

Chapter 7

VARIATIONAL METHODS

7.1 Introduction

The Galerkin method given earlier can be shown to produce element matrix integral definitions that would be identical to those obtained from an Euler variational form, if one exists. Most non-linear problems do not have a variational form, yet the Galerkin method and other weighted residual methods can still be used. Thus, one might ask, "Why consider variational methods?" There are several reasons for using them. One is that if the variational integral form is known, one does not have to derive the corresponding differential equation. Also, most of the important variational statements for problems in engineering and physics have been known for over 200 years. Another important feature of variational methods is that often dual principles exist that allow one to establish both an upper bound estimate and a lower bound estimate for an approximate solution. These can be very helpful in establishing accurate error estimates for adaptive solutions. Thus, the variational methods still deserve serious study, especially the energy methods of solid mechanics.

We have seen that the weighted residual methods provide several approaches for generating approximate (or exact) solutions based on equivalent integral formulations of the original partial differential equations. *Variational Methods*, or the *Calculus of Variations* have given us another widely used set of tools for equivalent integral formulations. They were developed by the famous mathematician Euler in the mid-1700's. Since that time the variational forms of most elliptic partial differential equations have been known. It has been proved that a variational form and the Galerkin method yield the same integral formulations when a governing variational principal exists. Variational methods have thus been used to solve problems in elasticity, heat transfer, electricity, magnetism, ideal fluids, etc. Thus, it is logical to expect that numerical approximations based on these methods should be very fruitful. They have been very widely employed in elasticity and structural mechanics so we begin with that topic. Then we will introduce the finite element techniques as logical extensions of the various classical integral formulations.

7.2 Structural Mechanics

Modern structural analysis relies extensively on the finite element method. Its most popular integral formulation, based on the variational calculus of Euler, is the *Principal of Minimum Total Potential Energy*. (This is also known as the principal of virtual work.) Basically, it states that the displacement field that satisfies the essential displacement boundary conditions and minimizes the total potential energy is the one that corresponds to the state of static equilibrium. This implies that displacements are our primary unknowns. They will be interpolated in space as will their derivatives, the strains. The total potential energy, Π , is the strain energy, U , of the structure minus the mechanical work, W , done by the applied forces. From introductory mechanics that the mechanical work, W , done by a force is the scalar dot product of the force vector, \mathbf{F} , and the displacement vector, \mathbf{u} , at its point of application.

To illustrate the concept of energy formulations we will review the equilibrium of the well-known linear spring. Figure 7.2.1 shows a typical spring of stiffness k that has an applied force, F , at the free end. That end undergoes a displacement of Δ . The work done by the single force is

$$W = \vec{\Delta} \cdot \vec{F} = \Delta F. \quad (7.1)$$

The spring stores potential energy due to its deformation. Here we call that strain energy. That energy is given by

$$U = \frac{1}{2} k \Delta^2. \quad (7.2)$$

Therefore, the total potential energy for the loaded spring is

$$\Pi(\Delta) = U - W = \frac{1}{2} k \Delta^2 - \Delta F. \quad (7.3)$$

The equation of equilibrium is obtained by minimizing Π with respect to the displacement; that is, $\partial\Pi/\partial\Delta = 0$. This simplifies to the single scalar equation $k\Delta = F$, which is the well-known equilibrium equation for a linear spring. This example was slightly simplified, since we started with the condition that the left end of the spring

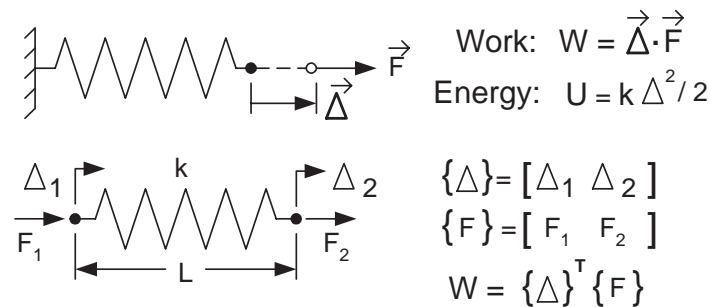


Figure 7.2.1 A simple linear spring

had no displacement (an essential boundary condition). Next we will consider a spring where either end can be fixed or free to move.

The elastic bar is often modeled as a linear spring. In introductory mechanics of materials the axial stiffness of a bar is defined as $k = EA/L$ where it has a length of L , an area A , and an elastic modulus of E . Our spring model of the bar (see Fig. 7.2.1) has two end displacements, Δ_1 and Δ_2 , and two associated axial forces, F_1 and F_2 . The net deformation of the bar is $\Delta = \Delta_2 - \Delta_1$. We denote the total vector of displacements as $\mathbf{D}^T = [\Delta_1 \ \Delta_2]$ and the associated vector of forces as $\mathbf{F}^T = [F_1 \ F_2]$. Then the work done on the bar is

$$W = \mathbf{D}^T \mathbf{F} = \Delta_1 F_1 + \Delta_2 F_2.$$

The net displacement will be expressed in matrix form here to compare with the later mathematical formulations. It is $\Delta = [-1 \ 1] \mathbf{D}$. Then the spring's strain energy is

$$U = \frac{1}{2} k \Delta^2 = \frac{EA}{2L} \mathbf{D}^T \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} [-1 \ 1] \mathbf{D} = \frac{1}{2} \mathbf{D}^T \mathbf{K} \mathbf{D}$$

where the bar stiffness is

$$\mathbf{K} = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The total potential energy, Π , depends on all the displacements, \mathbf{D} :

$$\Pi(\mathbf{D}) = \frac{1}{2} \mathbf{D}^T \mathbf{K} \mathbf{D} - \mathbf{D}^T \mathbf{F} \quad (7.4)$$

and the equation of equilibrium comes from the minimization

$$\partial \Pi / \partial \mathbf{D} = \mathbf{0}, \quad \text{or} \quad \mathbf{K} \mathbf{D} = \mathbf{F} \quad (7.5)$$

represents the system of algebraic equations of equilibrium for the elastic system. These two equations do not yet reflect the presence of an essential boundary condition, and $\det(\mathbf{K}) \equiv 0$ and the system is singular. These relations were developed on physical arguments and did not involve any finite element theory. Next we will see that a one-dimensional FEA yields the same forms.

7.3 Finite Element Analysis

Up to this point we have considered equivalent integral forms in the classical sense and not invoked their enhancement by finite element methods. We have seen that the resulting algebraic equation systems based on a global approximate solution are fully coupled. That is, the coefficient matrix is not *sparse* (not highly populated with zeros) so that the solution or inversion cost would be high. The finite element method lets us employ an integral form to obtain a set of sparse equations to be solved for the coefficients, \mathbf{D} , that yield the best approximation.

While we have looked so far mainly at one-dimensional problems in the general case we should be able to see that the residual error will involve volume integrals as well as surface integrals over part of the surface of the volume. For complicated shapes

encountered in solving practical problems it is almost impossible to assume a global solution that would satisfy the boundary conditions. Even if we could do that the computational expense would probably prevent us from obtaining a solution. Both of these important practical limitations can be overcome if we utilize a *piecewise approximation* that has only local support in space. That is part of what *finite element analysis* offers us.

The basic concept is that we split the actual solution domain into a series of sub-domains, or *finite elements*, that are interconnected in such a way that we can split the required integrals into a summation of integrals over all the element domains. If we restrict the approximation to a function that exists only within the element domain then the algebraic system becomes sparse because an element only directly interacts with those elements that are connected to it. By restricting the element to a single shape, or to a small library of shapes, we can do the required integrals over that shape and use the results repeatedly to build up the integral contributions over the entire solution domain. The main additional piece of work that results is the requirement that we do some bookkeeping to keep up with the contribution of each element. We refer to this as the equation *assembly*. That topic was illustrated in Fig. 1.3.1, and will be discussed in more detail below. In today's terminology the assembly procedure and the post-processing procedures are a series of *gather* and *scatter* operations.

Many finite element problems those concepts can be expressed symbolically in terms of a scalar quantity, I , such as

$$I(\mathbf{D}) = \frac{1}{2} \mathbf{D}^T \mathbf{K} \mathbf{D} + \mathbf{D}^T \mathbf{C} \rightarrow \min \quad (7.6)$$

where \mathbf{D} is a vector containing the unknown nodal parameters associated with the problem, and \mathbf{K} and \mathbf{C} are matrices defined in terms of the element properties and geometry. The above quantity is known as a *quadratic form*. If one uses a variational formulation then the solution of the finite element problem is usually required to satisfy the following system equations: $\partial I / \partial \mathbf{D} = \mathbf{0}$. In the finite element analysis one assumes that the (scalar) value of I is given by the sum of the element contributions. That is, one assumes

$$I(\mathbf{D}) = \sum_{e=1}^{n_e} I^e(\mathbf{D})$$

where I^e is the contribution of element number 'e'. One can (but does not in practice) define I^e in terms of \mathbf{D} such that

$$I^e(\mathbf{D}) = \frac{1}{2} \mathbf{D}^T \mathbf{K}^e \mathbf{D} + \mathbf{D}^T \mathbf{C}^e \quad (7.7)$$

where the \mathbf{K}^e are the same size as \mathbf{K} , but very sparse. Therefore, Eq. 7.7 is

$$I(\mathbf{D}) = \frac{1}{2} \mathbf{D}^T \left(\sum_{e=1}^{n_e} \mathbf{K}^e \right) \mathbf{D} + \mathbf{D}^T \left(\sum_{e=1}^{n_e} \mathbf{C}^e \right) \quad (7.8)$$

and comparing this with Eq. 7.6 one can identify the relations

$$\mathbf{K} = \sum_{e=1}^{n_e} \mathbf{K}^e, \quad \mathbf{C} = \sum_{e=1}^{n_e} \mathbf{C}^e. \quad (7.9)$$

If n_d represents the total number of unknowns in the system, then the size of these

matrices are $n_d \times n_d$ and $n_d \times 1$, respectively.

As a result of Eq. 7.9 one often sees the statement, "the system matrices are simply the sum of the corresponding element matrices." This is true, and indeed the symbolic operations depicted in the last equation are simple but one should ask (while preparing for the ensuing handwaving), "in practice, how are the element matrices obtained and how does one carry out the summations?" Before attempting to answer this question, it will be useful to backtrack a little. First, it has been assumed that an element's behavior, and thus its contribution to the problem, depends *only* on those nodal parameters that are associated with the element. In practice, the number of parameters associated with a single element usually lies between a minimum of 2 and a maximum of 96; with the most common range in the past being from three to eight. (However, for hierarchical elements in three-dimensions it is possible for it to be 27!) By way of comparison, n_d can easily reach a value of several thousand, or several hundred thousand. Consider an example of a system where $n_d = 5000$. Let this system consist of one-dimensional elements with two parameters per element. A typical matrix \mathbf{C}^e will contain 5000 terms and all but two of these terms will be identically zero since only those two terms of \mathbf{D} , 5000×1 , associated with element 'e' are of any significance to element 'e'. In a similar manner one concludes that, for the present example, only four of the 25,000,000 terms of \mathbf{K}^e would not be identically zero. Therefore, it becomes obvious that the symbolic procedure introduced here is not numerically efficient and would not be used in practice. There are some educational uses of the symbolic procedure that justify pursuing it a little further. Recalling that it is assumed that the element behavior depends only on those parameters, says ϕ^e , that are associated with element 'e', it is logical to assume that

$$I^e = \frac{1}{2} \phi^{eT} \mathbf{k}^e \phi^e + \phi^{eT} \mathbf{c}^e. \quad (7.10)$$

If n_i represents the number of degrees of freedom associated with the element then the element vectors ϕ^e and \mathbf{c}^e are $n_i \times 1$ in size and the size of the square matrix \mathbf{k}^e is $n_i \times n_i$. Note that in practice n_i is usually much less than n_d , but they can be the equal. The matrices \mathbf{k}^e and \mathbf{c}^e are commonly known as *the* element matrices. For the one-dimensional element discussed in the previous example, \mathbf{k}^e and \mathbf{c}^e would be 2×2 and 2×1 in size, respectively, and would represent the only coefficients in \mathbf{K}^e and \mathbf{C}^e that are not identically zero.

All that remains is to relate \mathbf{k}^e to \mathbf{K}^e and \mathbf{c}^e to \mathbf{C}^e . Obviously Eqs. 7.7 and 7.10 are equal and are the key to the desired relations. In order to utilize these equations, it is necessary to relate the degrees of freedom of the element ϕ^e , to the degrees of freedom of the total system \mathbf{D} . This is done symbolically by introducing a $n_i \times n_d$ bookkeeping matrix, β^e , such that the following identity is satisfied:

$$\phi^e \equiv \beta^e \mathbf{D}. \quad (7.11)$$

Substituting this identity, Eq. 7.10 is expressed in terms of the system level unknowns as $I^e(\mathbf{D}) = \frac{1}{2} \mathbf{D}^T (\beta^{eT} \mathbf{k}^e \beta^e) \mathbf{D} + \mathbf{D}^T \beta^{eT} \mathbf{c}^e$. Comparing this relation with Eq. 7.7, one can establish the symbolic relationships $\mathbf{K}^e = \beta^{eT} \mathbf{k}^e \beta^e$, $\mathbf{C}^e = \beta^{eT} \mathbf{c}^e$ and denote the assembly process by the symbol \mathbf{A}_e so

$$\mathbf{K} = \sum_{e=1}^{n_e} \boldsymbol{\beta}^{eT} \mathbf{k}^e \boldsymbol{\beta}^e = \mathbf{A} \mathbf{K}^e, \quad \mathbf{C} = \sum_{e=1}^{n_e} \boldsymbol{\beta}^{eT} \mathbf{c}^e = \mathbf{A} \mathbf{C}^e. \quad (7.12)$$

Equation 7.12 can be considered as the symbolic definitions of the *assembly operator* and its procedures relating *the* element matrices, \mathbf{k}^e and \mathbf{c}^e , to the total system matrices, \mathbf{K} and \mathbf{C} . Note that these relations involve the element connectivity (topology), $\boldsymbol{\beta}^e$, as well as the element behavior, \mathbf{k}^e and \mathbf{c}^e . Although some programs do use this procedure, it is very inefficient and thus very expensive.

For the sake of completeness, the $\boldsymbol{\beta}^e$ matrix will be briefly considered. To simplify the discussion, it will be assumed that each nodal point has only a single unknown scalar nodal parameter (degree of freedom). Define a mesh consisting of four triangular elements. Figure 7.3.1 shows both the system and element degree of freedom numbers. The system degrees of freedom are defined as $\mathbf{D}^T = [\Delta_1 \ \Delta_2 \ \Delta_3 \ \Delta_4 \ \Delta_5 \ \Delta_6]$ and the degrees of freedom of element ‘ e ’ are $\boldsymbol{\phi}^{eT} = [\phi_1 \ \phi_2 \ \phi_3]^e$. The connectivity or topology data supplied for these elements are also shown in that figure. Thus, for element number four ($e = 4$), these quantities are related by

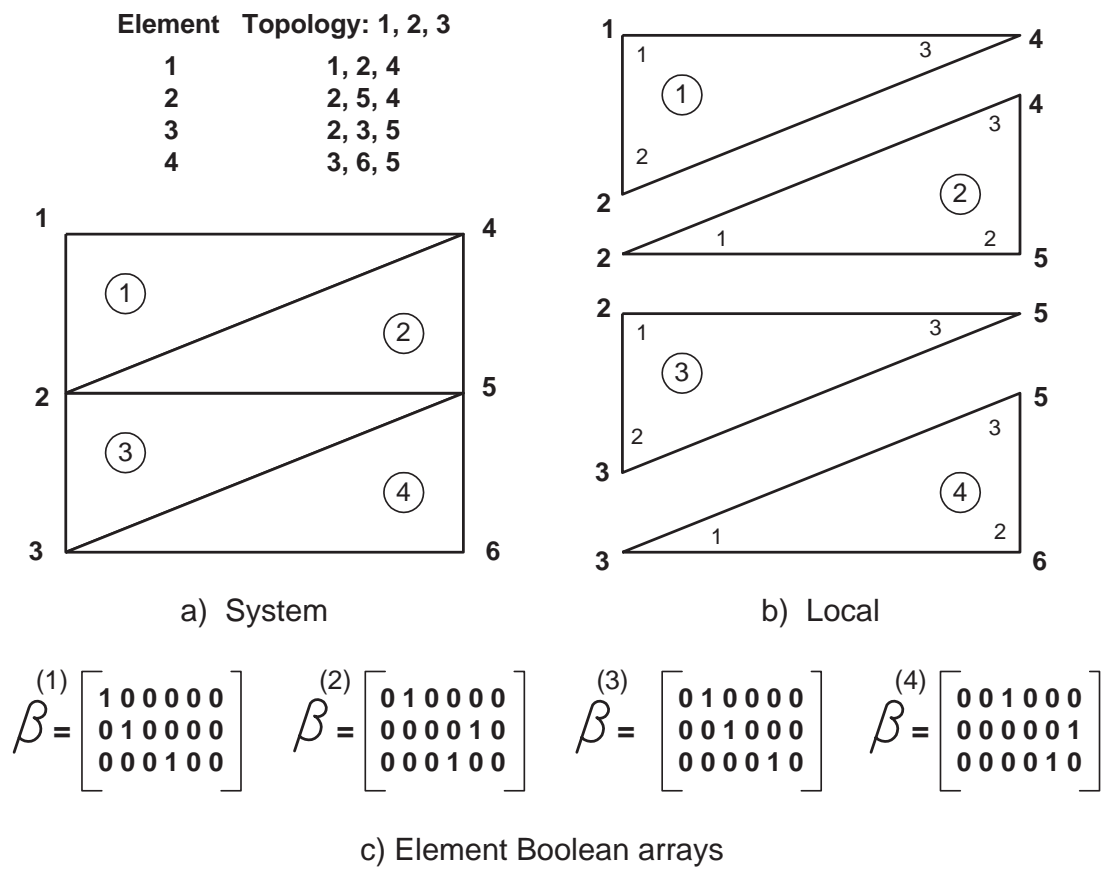


Figure 7.3.1 Relationship between system and element degrees of freedom

$$\boldsymbol{\phi}^{(4)T} = [\phi_1 \quad \phi_2 \quad \phi_3]^{(4)} = [\Delta_3 \quad \Delta_6 \quad \Delta_5]$$

which can be expressed as the gather operation $\boldsymbol{\phi}^{(4)} \equiv \boldsymbol{\beta}^{(4)}\Delta$ where

$$\boldsymbol{\beta}^{(4)} \equiv \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The matrices $\boldsymbol{\beta}^{(1)}$, $\boldsymbol{\beta}^{(2)}$, and $\boldsymbol{\beta}^{(3)}$ can be defined in a similar manner, and are given in the figure. Since the matrix $\boldsymbol{\beta}^e$ contains only ones and zeros, it is called the element Boolean or binary matrix. Note that there is a single unity term in each row and all other coefficients are zero. Therefore, this $n_i \times n_d$ array will contain only n_i non-zero (unity) terms, and since $n_i \ll n_d$, the matrix multiplications of any Boolean gather or scatter operation are numerically very inefficient. There is a common shorthand method for writing any Boolean matrix to save space. It is written as a vector with the column number that contains the unity term on any row. In that form we would write $\boldsymbol{\beta}^{(4)}$ as

$$\boldsymbol{\beta}^{(4)T} \leftrightarrow [3 \quad 6 \quad 5]$$

which you should note is the same as the element topology list because there is only one parameter per node. If we had formed the example with two parameters per node ($n_g = 2$) then the Boolean array would be

$$\boldsymbol{\beta}^{(4)T} \leftrightarrow [5 \quad 6 \quad 11 \quad 12 \quad 9 \quad 10].$$

This more compact vector mode was used in the assembly figures in Chap. 1. There it was given the array name *INDEX*.

The transpose of a $\boldsymbol{\beta}$ matrix can be used to scatter the element terms into the system vector. For element four we see that $\boldsymbol{\beta}^{(4)T} \mathbf{c}^{(4)} = \mathbf{C}^{(4)}$ gives

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \end{Bmatrix}^{(4)} = \begin{Bmatrix} 0 \\ 0 \\ C_1 \\ 0 \\ C_2 \\ C_3 \end{Bmatrix}^{(4)}.$$

This helps us to see how the scatter and sum operation for \mathbf{C} in Eq. 7.12 actually works. The Boolean arrays, $\boldsymbol{\beta}$, have other properties that are useful in understanding certain other element level operations. For future reference note that $\boldsymbol{\beta}_e \boldsymbol{\beta}_e^T = \mathbf{I}$, and that if elements i and j are not connected $\boldsymbol{\beta}_i \boldsymbol{\beta}_j^T = \mathbf{0} = \boldsymbol{\beta}_j \boldsymbol{\beta}_i^T$, and if they are connected then $\boldsymbol{\beta}_i \boldsymbol{\beta}_j^T = \mathbf{X}_{ij}$ where \mathbf{X}_{ij} indicates the *dof* that are common to both. That is, the Boolean array \mathbf{X} is zero except for those *dof* common to both element i and j .

Although these symbolic relations have certain educational uses their gross inefficiency for practical computer calculations led to the development of the equivalent programming procedure of the "direct method" of assembly that was discussed earlier, and illustrated in Figs. 1.3.2 and 3. It is useful at times to note that the identity (7.11) leads to the relation

$$\frac{\partial(\)^e}{\partial \mathbf{D}} = \boldsymbol{\beta}^{eT} \frac{\partial(\)^e}{\partial \boldsymbol{\phi}^e}, \quad (7.13)$$

where $(\)^e$ is some quantity associated with element 'e'. At this point we will begin to illustrate finite element domains and their piecewise local polynomial approximations to variational approximations by applying them to an elastic rod.

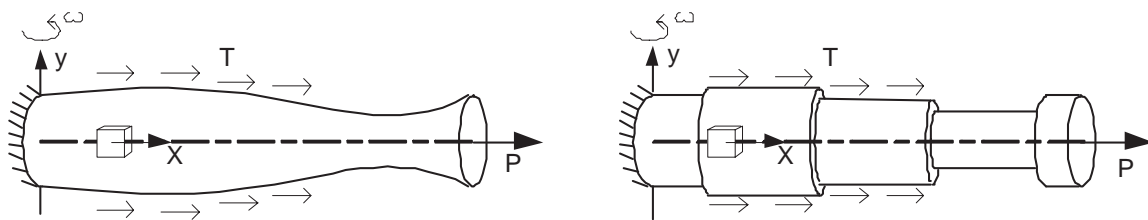
Before leaving the assembly relations for a while one should consider their extension to the case where there are more than one unknown per node. This was illustrated in Fig. 1.3.4 for three line elements with two nodes each and two dof per node. There the connectivity data and corresponding equation numbers are also given and we note that the connections between local and global equation numbers occur in pairs. Now we are inserting groups of two by two submatrices into the larger system matrix. It is not unusual for six dof to occur at each node. Then we assemble six by six blocks.

7.4 Continuous Elastic Bar

Consider an axisymmetric rod shown in Fig. 7.4.1. The cross-sectional area, $A(x)$, the perimeter, $p(x)$, the material modulus of elasticity, $E(x)$, and axial loading conditions would, in general, depend on the axial coordinate, x . The loading conditions could include surface tractions (shear) per unit area, $T(x)$, body forces per unit volume, $X(x)$, and concentrated point loads, P_i at point i . The axial displacement at a point will be denoted by $u(x)$, and its value at point i is u_i . The work done by a force is the product of the force, the displacement at the point of application of the force, and the cosine of the angle between the force and the displacement. Here the forces are all parallel so the cosine is either plus or minus one. Evaluating the mechanical work

$$W = \int_0^L u(x) X(x) A(x) dx + \int_0^L u(x) T(x) p(x) dx + \sum_i u_i P_i. \quad (7.14)$$

As mentioned earlier the total potential energy, Π , includes the *strain energy*, U , and work of the externally applied forces, W . That is, $\Pi = U - W$. In a mechanics of materials course it is shown that the strain energy per unit volume is half the product of the stress and strain. The axial strain and stress will be denoted by $\varepsilon(x)$ and $\sigma(x)$,



X - Body force per unit volume, T - Traction force per unit area, P - Point load

Figure 7.4.1 A typical axially loaded bar

respectively. Thus, the strain energy is

$$U = \frac{1}{2} \int_0^L \sigma(x) \varepsilon(x) A(x) dx. \quad (7.15)$$

The latter two equations have used $dV = A dx$ and $dS = p dx$ where dS is an exterior surface area. The work is clearly defined in terms of the displacement, u , since the loads would be given quantities. For example, the body force could be gravity, $X = \rho(x) g$, or a centrifugal load due to rotation about the y -axis, $X = \rho(x) x \omega^2$. Surface tractions are less common in 1-D but it could be due to a very viscous fluid flowing over the outer surface and in the x -direction.

Our goal is to develop a displacement formulation. Thus, we also need to relate both the stress and strain to the displacement, u . We begin with the *strain-displacement relation* $\varepsilon(x) = du(x)/dx$ which relates the strain to the derivative of the displacement. The stress at a point is directly proportional to the strain at the point. Thus, it is also dependent on the *displacement gradient*. The relation between stress and strain is a *constitutive relation* known as *Hooke's law* $\sigma(x) = E(x) \varepsilon(x)$. Therefore, we now see that the total potential energy depends on the unknown displacements and displacement gradients. We are searching for the displacement configuration that minimizes the total potential energy since that configuration corresponds to the state of stable equilibrium. As we have suggested above, a finite element model can be introduced to approximate the displacements and their derivatives. Here, we begin by selecting the simplest model possible. That is, the two node line element with assumed linear displacement variation with x . As suggested, we now assume that each element has homogeneous properties and all the integrals can be represented as the sum of the corresponding element integrals (and the intersection of those elements with the boundary of the solution domain):

$$\Pi = \sum_{e=1}^{n_e} \Pi^e + \sum_{b=1}^{n_b} \Pi^b - \sum_i P_i u_i \quad (7.16)$$

where Π^e is the typical element domain contribution, and Π^b is the contribution from a typical boundary segment domain. This one-dimensional example is somewhat misleading since in general a surface traction, T , acts only on a small portion of the exterior boundary. Thus, the number of boundary domains, n_b , is usually much less than the number of elements, n_e . Here, however, $n_b = n_e$ and the distinction between the two may not become clear until two-dimensional problems are considered. Substituting Eqs. 7.14 and 7.15 into Eq. 7.3 and equating to Eq. 7.16 yields

$$\Pi^e = \frac{1}{2} \int_{L^e} \sigma^e \varepsilon^e A^e dx - \int_{L^e} u^e X^e A^e dx, \quad \Pi^b = - \int_{L^b} u^b T^b p^b dx$$

where $u^e(x)$ and $u^b(x)$ denote the approximated displacements in the element and on the boundary surface, respectively. In this special example, $u^b = u^e$ but that is not usually true. Symbolically we interpolate such that $u^e(x) = \mathbf{H}^e(x) \mathbf{u}^e = \mathbf{u}^{eT} \mathbf{H}^{eT}(x)$ and likewise if we degenerate this interpolation to a portion (or sub-set) of the boundary of the element $u^b(x) = \mathbf{H}^b(x) \mathbf{u}^b = \mathbf{u}^{bT} \mathbf{H}^{bT}(x)$. In the example we have the unusual case that $\mathbf{u}^b = \mathbf{u}^e$ and $\mathbf{H}^b = \mathbf{H}^e$. Generally \mathbf{u}^b is a subset of \mathbf{u}^e (i.e., $\mathbf{u}^b \subset \mathbf{u}^e$) and \mathbf{H}^b is a subset of \mathbf{H}^e . This interpolation relationship gives the strain approximation in an element:

$$\boldsymbol{\varepsilon}^e(x) = \frac{du^e}{dx} = \frac{d\mathbf{H}^e(x)}{dx} \mathbf{u}^e = \mathbf{u}^{eT} \frac{d\mathbf{H}^{eT}(x)}{dx}$$

or in more common notation

$$\boldsymbol{\varepsilon}^e = \mathbf{B}^e(x)\mathbf{u}^e \quad (7.17)$$

where here \mathbf{B}^e is called the "strain-displacement matrix" since it determines the mechanical strain from the element's nodal displacements. Likewise, the one-dimensional stress, $\boldsymbol{\sigma}$, is defined by "Hooke's Law", where \mathbf{E} is the modulus of elasticity (a material property) as:

$$\boldsymbol{\sigma}^e(x) = \mathbf{E}^e(x) \boldsymbol{\varepsilon}^e(x) \quad (7.18)$$

Substituting into the definition of the Total Potential Energy from the elements and boundary terms:

$$\begin{aligned} \Pi^e &= \frac{1}{2} \mathbf{u}^{eT} \int_{L^e} \mathbf{B}^{eT} E^e \mathbf{B}^e A^e \mathbf{u}^e dx - \mathbf{u}^{eT} \int_{L^e} \mathbf{H}^{eT} X^e A^e dx \\ \Pi^b &= -\mathbf{u}^{bT} \int_{L^b} \mathbf{H}^{bT} T^b p^b dx. \end{aligned}$$

The latter two relations can be written symbolically as

$$\Pi^e = \frac{1}{2} \mathbf{u}^{eT} \mathbf{S}^e \mathbf{u}^e - \mathbf{u}^{eT} \mathbf{C}_x^e, \quad \Pi^b = -\mathbf{u}^{bT} \mathbf{C}_T^b \quad (7.19)$$

where the element stiffness matrix is

$$\mathbf{S}^e = \int_{L^e} \mathbf{B}^{eT} E^e \mathbf{B}^e A^e dx, \quad (7.20)$$

The vast majority of finite element problems have at least one square matrix of this form, that involves the matrix product $\mathbf{B}^{eT} \mathbf{E} \mathbf{B}^e$. We will see later that calculation of the element error estimator also requires the use of the \mathbf{E}^e and \mathbf{B}^e arrays. Thus, even if the \mathbf{S}^e matrix is simple enough to write in closed form there are other reasons why we may want to form the \mathbf{E}^e and \mathbf{B}^e arrays at the same time. The element body force vector is

$$\mathbf{C}_x^e = \int_{L^e} \mathbf{H}^{eT} X^e A^e dx, \quad (7.21)$$

and the boundary segment traction vector is

$$\mathbf{C}_T^b = \int_{L^b} \mathbf{H}^{bT} T^b p^b dx. \quad (7.22)$$

The Total Potential Energy of the system is

$$\Pi = \frac{1}{2} \sum_e \mathbf{u}^{eT} \mathbf{S}^e \mathbf{u}^e - \sum_e \mathbf{u}^{eT} \mathbf{C}_x^e - \sum_b \mathbf{u}^{bT} \mathbf{C}_T^b - \mathbf{u}^T \mathbf{P} \quad (7.23)$$

where \mathbf{u} is the vector of all of the unknown nodal displacements. Here we have assumed that the external point loads are applied at node points only. The last term represents the scalar, or dot, product of the nodal displacement and nodal forces. That is,

$$\mathbf{u}^T \mathbf{P} = \mathbf{P}^T \mathbf{u} = \sum_i u_i P_i.$$

Of course, in practice most of the P_i are zero. By again applying the direct assembly procedure, or from the Boolean assembly operations, the Total Potential Energy is $\Pi(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{S} \mathbf{u} - \mathbf{u}^T \mathbf{C}$ and minimizing with respect to all the unknown displacements, \mathbf{u} , gives the algebraic equilibrium equations for the entire structure $\mathbf{S} \mathbf{u} = \mathbf{C}$. Therefore, we see that our variational principle has lead to a very general and powerful formulation for this class of structures. It automatically includes features such as variable material properties, variable loads, etc. These were difficult to treat when relying solely on physical intuition. Although we will utilize the simple linear element none of our equations are restricted to that definition of \mathbf{H} and the above symbolic formulation is valid for any linear elastic solid of any shape. If we substitute \mathbf{H}^e for the linear element

$$\mathbf{H}^e(x) = \begin{bmatrix} \frac{(x_2^e - x)}{L^e} & \frac{(x - x_1^e)}{L^e} \end{bmatrix},$$

then

$$\mathbf{B}^e = \frac{d\mathbf{H}^e}{dx} = \begin{bmatrix} -\frac{1}{L^e} & \frac{1}{L^e} \end{bmatrix}.$$

and assume constant properties ($\mathbf{E}^e, \mathbf{A}^e$), then the element and boundary matrices are simple to integrate. The results are

$$\mathbf{S}^e = \frac{E^e A^e}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

$$\mathbf{C}_x^e = \frac{X^e A^e L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}, \quad \mathbf{C}_T^e = \frac{T^e P^e L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

Also in this case one obtains the strain-displacement relation

$$\boldsymbol{\varepsilon}^e = \frac{1}{L^e} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1^e \\ u_2^e \end{Bmatrix} \quad (7.24)$$

which means that the strain is constant in the element but the displacement approximation is linear. It is common to refer to this element as the constant strain line element, CSL. The above stiffness matrix is the same as that obtained in Sec. 7.1. The load vectors take the resultant element, or boundary, force and place half at each node. That logical result does not carry over to more complicated load conditions, or interpolation functions and it then becomes necessary to rely on the mathematics of Eqs. 7.21 and 7.22. The implementation of this element will be given after thermal loading is defined.

As an example of a slightly more difficult loading condition consider a case where the body force varies linearly with x . This could include the case of centrifugal loading mentioned earlier. For simplicity assume a constant area A and let us define the value of the body force at each node of the element. To define the body force at any point in the element we again utilize the interpolation function and set

$$X^e(x) = \mathbf{H}^e(x) \mathbf{X}^e \quad (7.25)$$

where \mathbf{X}^e are the defined nodal values of the body force. For these assumptions the body force vector becomes

$$\mathbf{C}_x^e = A^e \int_{L^e} \mathbf{H}^{eT} \mathbf{H}^e dx \mathbf{X}^e .$$

For the linear element the integration reduces to

$$\mathbf{C}_x^e = \frac{A^e L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} X_1^e \\ X_2^e \end{Bmatrix} .$$

This agrees with our previous result for constant loads since if $X_1^e = X_2^e = X^e$, then

$$\mathbf{C}_x^e = \frac{A^e L^e X^e}{6} \begin{Bmatrix} 2+1 \\ 1+2 \end{Bmatrix} = \frac{A^e L^e X^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} .$$

A more common problem is that illustrated in Fig. 7.4.1 where the area of the member varies along the length. To approximate that case, with constant properties, one could interpolate for the area at any point as $A^e(x) = \mathbf{H}^e(x) \mathbf{A}^e$, then the stiffness in Eq. 7.20 becomes

$$\mathbf{S}^e = \frac{E^e}{(L^e)^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} V^e$$

where

$$V^e = \int_{L^e} A^e dx = \int_{L^e} \mathbf{H}^e(x) dx \mathbf{A}^e = \frac{L^e}{2} [1 \quad 1] \begin{Bmatrix} A_1^e \\ A_2^e \end{Bmatrix} = \frac{L^e (A_1^e + A_2^e)}{2}$$

is the average volume of the element. Using that volume value the body force vector is

$$\mathbf{C}_x^e = \frac{X^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} V^e = \frac{X^e L^e (A_1^e + A_2^e)}{4} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} ,$$

but if we assume a constant body force per unit volume and interpolate for the local area we get

$$\mathbf{C}_x^e = X^e \int_{L^e} \mathbf{H}^{eT} \mathbf{H}^e dx \mathbf{A}^e = \frac{X^e L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} A_1^e \\ A_2^e \end{Bmatrix} .$$

The above approximations should be reasonably accurate. However, for a cylindrical bar the area is related to the radius by $A = \pi r^2$. Thus, it would be slightly more accurate to describe the radius at each end and interpolate $r^e(x) = \mathbf{H}^e(x) \mathbf{r}^e$ so that

$$\begin{aligned} V^e &= \int_{L^e} A^e dx = \pi \int_{L^e} r^e(x)^2 dx = \mathbf{r}^{eT} \pi \int_{L^e} \mathbf{H}^{eT} \mathbf{H}^e dx \mathbf{r}^e \\ &= \mathbf{r}^{eT} \frac{\pi L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{r}^e = \pi L^e (r_1^2 + r_1 r_2 + r_2^2) / 3 . \end{aligned}$$

Of course, for a bar of constant radius (and area) all three approaches give identical resultant body force components at the nodes.

Clearly, as one utilizes more advanced interpolation functions, the integrals involved in Eqs. 7.20 to 7.22 become more difficult to evaluate. An example to illustrate the use of these element matrices and to introduce the benefits of post-solution calculations follows. Consider a prismatic bar of steel rigidly fixed to a bar of brass and subjected to a vertical load of $P = 10,000$ lbs., as shown in Fig. 7.4.2. The structure is supported at the top point and is also subjected to a gravity (body force) load. We wish to determine the deflections, reactions, and stresses for the properties :

Element	L^e	A^e	E^e	X^e	Topology	
1	420"	10 sq. in.	30×10^6 psi	0.283 lb/in^3	1	2
2	240"	8 sq. in.	13×10^6 psi	0.300 lb/in^3	2	3

The first element has an axial stiffness constant of $EA/L = 0.7143 \times 10^6$ lb/in. and the body force is $XAL = 1,188.6$ lbs. while for the second element the corresponding terms

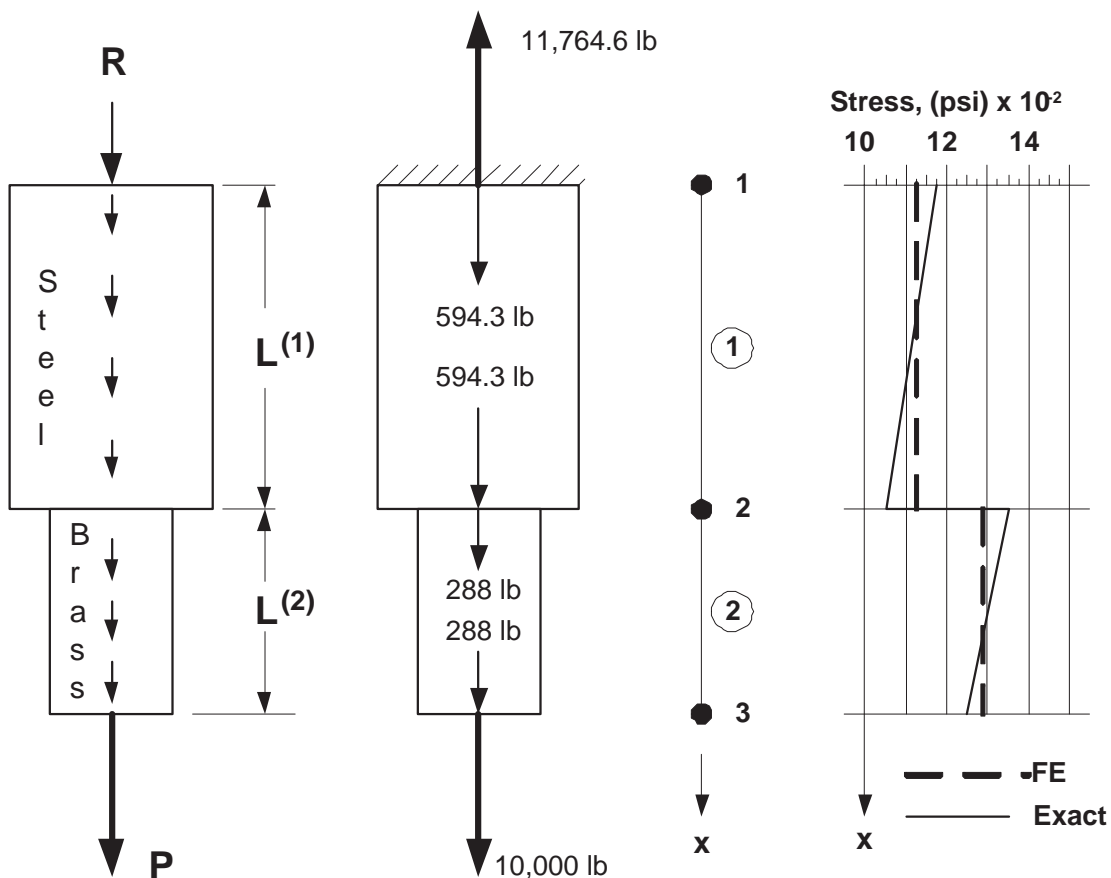


Figure 7.4.2 An axially loaded system

```

title "Steel-Brass gravity and end load" ! begin keywords ! 1
nodes 3 ! Number of nodes in the mesh ! 2
elems 2 ! Number of elements in the system ! 3
dof 1 ! Number of unknowns per node ! 4
el_nodes 2 ! Maximum number of nodes per element ! 5
space 1 ! Solution space dimension ! 6
b_rows 1 ! Number of rows in the B (operator) matrix ! 7
shape 1 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex ! 8
remarks 2 ! Number of user remarks ! 9
el_real 5 ! Number of real properties per element !10
loads ! External source supplied !11
el_react ! Compute & list element reactions !12
post_el ! Require post-processing, create n_file1 !13
quit ! keyword input, remarks follow !14
Nodal displacements are exact, stress exact only at center !15
Properties: A, E, DT, ALPHA, GAMMA !16
1 1 0.00 ! begin nodes !17
2 0 420. !18
3 0 660. !19
1 1 2 ! begin elements !20
2 2 3 !21
1 1 0. ! essential bc !22
1 10. 30.e6 0. 0. 0.283 ! el, A, E, DT, ALPHA, GAMMA !23
2 8. 13.e6 0. 0. 0.300 ! el, A, E, DT, ALPHA, GAMMA !24
3 1 1.e4 ! node, direction, load (stop with last) !25

```

Figure 7.4.3 The steel-brass bar example typical input

```

*** REACTION RECOVERY *** ! 1
NODE, PARAMETER, REACTION, EQUATION ! 2
1, DOF_1, -1.1765E+04 1 ! 3
! 4
*** OUTPUT OF RESULTS IN NODAL ORDER *** ! 5
NODE, X-Coord, DOF_1, ! 6
1 0.0000E+00 0.0000E+00 ! 7
2 4.2000E+02 1.5638E-02 ! 8
3 6.6000E+02 3.9380E-02 ! 9
!10
** ELEMENT REACTION, INTERNAL SOURCES AND SUMMATIONS ** !11
ELEMENT 1 !12
NODE DOF REACTION ELEM_SOURCE SUMS !13
1 1 -1.17646E+04 5.94300E+02 !14
2 1 1.05760E+04 5.94300E+02 !15
SUM: 1 -1.18860E+03 1.18860E+03 0.00000E+00 !16
ELEMENT 2 !17
NODE DOF REACTION ELEM_SOURCE SUMS !18
2 1 -1.05760E+04 2.88000E+02 !19
3 1 1.00000E+04 2.88000E+02 !20
SUM: 1 -5.76000E+02 5.76000E+02 0.00000E+00 !21
!22
ELEMENT STRESSES !23
ELEMENT STRESS MECH. STRAIN THERMAL STRAIN !24
1 1.11703E+03 3.72343E-05 0.00000E+00 !25
2 1.28600E+03 9.89231E-05 0.00000E+00 !26

```

Figure 7.4.4 The steel-brass bar example selected output

are 0.4333×10^6 lb/in. and 576 lbs., respectively. The system nodal force vector is $P^T = [R \ 0 \ 10,000]^{\text{lb}}$. Where R is the unknown reaction at node 1. Assembling the equations gives

$$10^5 \begin{bmatrix} 7.143 & -7.143 & 0 \\ -7.143 & 7.143 + 4.333 & -4.333 \\ 0 & -4.333 & 4.333 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} R \\ 0 \\ 10,000 \end{Bmatrix} + \frac{1}{2} \begin{Bmatrix} 1,188.6 \\ 1,188.6 + 576. \\ 576. \end{Bmatrix} = \begin{Bmatrix} R + 594.3 \\ 882.3 \\ 10,288 \end{Bmatrix}.$$

Applying the essential condition that $u_1 = 0$

$$10^5 \begin{bmatrix} 11.476 & -4.333 \\ -4.333 & 4.333 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} 882.3 \\ 10,288 \end{Bmatrix}$$

so that $u_2 = 1.5638 \times 10^{-2}$ in., $u_3 = 3.9381 \times 10^{-2}$ in., and determining the reaction from the first system equation: $R = -11,764.6$ lb. This reaction is compared with the applied loads in Fig. 7.4.2b. Now that all the displacements are known we can *post-process* the results to determine the other quantities of interest. Substituting into the element strain-displacement relation, Eq. 7.24 gives

$$\boldsymbol{\varepsilon}^{(1)} = \frac{1}{420} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{Bmatrix} 0.0 \\ 0.01564 \end{Bmatrix} = 3.724 \times 10^{-5} \text{ in/in}$$

$$\boldsymbol{\varepsilon}^{(2)} = \frac{1}{240} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{Bmatrix} 0.01564 \\ 0.03938 \end{Bmatrix} = 9.892 \times 10^{-5} \text{ in/in}$$

and from Eq. 7.18 the element stresses are

$$\boldsymbol{\sigma}^{(1)} = \mathbf{E}^{(1)} \boldsymbol{\varepsilon}^{(1)} = 30 \times 10^6 (7.724 \times 10^{-5}) = 1,117 \text{ lb/in}^2$$

$$\boldsymbol{\sigma}^{(2)} = \mathbf{E}^{(2)} \boldsymbol{\varepsilon}^{(2)} = 13 \times 10^6 (9.892 \times 10^{-5}) = 1,286 \text{ lb/in}^2.$$

These approximate stresses are compared with the exact stresses in Fig. 7.4.2d. This suggests that if accurate stresses are important then more elements are required to get good estimates from the piecewise constant element stress approximations. Note that the element stresses are exact if they are considered to act only at the element center. The input data and selected results for this example are given in Fig. 7.4.3 and 4, respectively.

7.5 Thermal Loads on a Bar

Before leaving the bar element it may be useful to note that another common loading condition can be included, that is the loading due to an initial thermal strain, so $\boldsymbol{\varepsilon} = \boldsymbol{\sigma}/E + \boldsymbol{\varepsilon}_t$. The thermal strain, $\boldsymbol{\varepsilon}_t$, due to a temperature rise of Δt is $\boldsymbol{\varepsilon}_t = \alpha \Delta t$ where α is the coefficient of thermal expansion. The work term in Eq. 7.14 is extended to include this effect by adding an *initial strain* contribution

$$W_t = \int_0^L \boldsymbol{\sigma}^T \boldsymbol{\varepsilon}_t A(x) dx.$$

This defines an element thermal force vector

$$\mathbf{C}_t^e = \int_{L^e} \mathbf{B}^{eT}(x) E^e(x) \alpha^e(x) \Delta t(x) A^e(x) dx$$

or for constant properties and uniform temperature rise

$$\mathbf{C}_t^{eT} = E^e \alpha^e \Delta t^e A^e \begin{bmatrix} -1 & +1 \end{bmatrix}. \quad (7.27)$$

There is a corresponding change in the constitutive law such that $\boldsymbol{\sigma} = \mathbf{E}(\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_t)$.

The coding requires a few new interface items listed in Fig. 7.5.1. The source code for implementing the linear elastic bar element is given in Fig. 7.5.2. There the last action, in line 36, is to save the modulus of elasticity and the strain data for later post-processing. The data keyword "post_el" activates the necessary sequential storage unit (n_file1). After the unknowns have been computed we gather typical data back to the element for use in post-processing secondary items in the element. This too requires a few interface items to the MODEL program and they are listed in Fig. 7.5.3. The stress recovery coding is given in Fig. 7.5.4 and it is invoked by the presence of the same 'post_el' data keyword. Since strain, initial strain, and stress are tensor quantities that have several components, stored in subscripted arrays, a unit subscript is required to remind us that we are dealing with their one-dimensional form. Up to this point we have dealt with scalars. The mechanical strain is found in line 25 and the generalized Hooke's Law is employed, at line 28, to recover the stress.

As a numerical example of this loading consider the previous model with the statically indeterminate supports in Fig. 7.5.5. The left support is fixed but the right support is displaced to the left 0.001 in. and the system is cooled by 35° F. Find the stress in each member if $\alpha_1 = 6.7 \times 10^{-6}$ and $\alpha_2 = 12.5 \times 10^{-6}$ in/in F. The assembled equations are

$$10^5 \begin{bmatrix} 7.143 & -7.143 & 0 \\ -7.143 & 11.476 & -4.333 \\ 0 & -4.333 & 4.333 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = 10^4 \begin{Bmatrix} -7.035 \\ 7.035 - 4.550 \\ +4.550 \end{Bmatrix} + \begin{Bmatrix} P_1 \\ 0 \\ P_2 \end{Bmatrix}$$

applying the boundary conditions that $u_1 = 0$, and $u_3 = -0.001$ in. and solving for u_2 yields $u_2 = -0.02203$ in. The reactions at points 1 and 3 are $P_1 = -54,613$ lb., and $P_3 = +54,613$ lb. Substituting into the element strain-displacement matrices yields element mechanical strains of -5.245×10^{-5} in/in., and $+8.763 \times 10^{-5}$ in/in., respectively. But, the initial thermal strains were -2.345×10^{-4} , and -4.375×10^{-4} , respectively so together they result in net tensile stresses in the elements of 5.46×10^3 and 6.83×10^3 psi, respectively. That is, the tension due to the temperature reduction exceeds the compression due to the support movement. Sample input and selected outputs for the above thermal loading example are given in Figs. 7.5.6 and 7, respectively. Note that the outputs for the last two examples have included the element reactions, also called the element level flux balances. These are often physically important so we will

```

                Interface from MODEL to ELEM_SQ_MATRIX, 2
Type   Status Name   Remarks                                     (keyword)
INTEGER (IN) IE      Current element number
INTEGER (IN) LT_FREE Number of element type unknowns
INTEGER (IN) LT_N    Number of element type solution nodes
INTEGER (IN) N_R_B   Number of rows in B and E arrays (b_rows)
INTEGER (IN) N_FILE1 Optional user sequential unit (post_el)

REAL(DP) (OUT) B (N_R_B, LT_FREE) Gradient-dof transformation
REAL(DP) (OUT) DLH (LT_PARM, LT_N) Local parametric derivatives of H
REAL(DP) (OUT) E (N_R_B, N_R_B)   Constitutive array
REAL(DP) (OUT) H_INTG (LT_N)      Integral of H array

REAL(DP) automatic XYZ (N_SPACE)  Coordinates of a point

GET_H_AT_QP          Form H array at quadrature point
GET_REAL_LP (k)      Gather real property k for current element
                    1 <= k <= (el_real)

```

Figure 7.5.1 User interface to ELEM_SQ_MATRIX, part 2

```

! ..... ! 1
! *** ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW *** ! 2
! ..... ! 3
! Define any new local array or variable types, then statements ! 4
! ..... ! 5
! AN AXIAL BAR BY DIRECT ENERGY APPROACH ! 6
! ELEMENT REAL PROPERTIES: (1) = AREA, (2) = ELASTIC MODULUS ! 7
! (3) = TEMP RISE, (4) = COEFF EXPANSION, (5) = WEIGHT DENSITY ! 8
! ..... ! 9
REAL(DP) :: BAR_L ! length !10
REAL(DP) :: DELTA_T, ALPHA ! temp rise, expansion !11
REAL(DP) :: AREA, GAMMA ! area, wt. density !12
REAL(DP) :: M_E, THERMAL ! modulus, thermal strain !13
! ..... !14
! Get properties for this element, IE !15
AREA = GET_REAL_LP (1); M_E = GET_REAL_LP (2) !16
DELTA_T = GET_REAL_LP (3); ALPHA = GET_REAL_LP (4) !17
GAMMA = GET_REAL_LP (5) !18
! ..... !19
! Find bar length and direction cosines !20
BAR_L = COORD (2, 1) - COORD (1, 1) ! length !21
! ..... !22
! Form global strain-displacement matrix !23
B (1, :) = (/ -1, 1 /) / BAR_L !24
! ..... !25
! Form global stiffness, S = B' EAL B !26
S = M_E * AREA * BAR_L * MATMUL ( TRANSPOSE (B), B ) !27
! ..... !28
! Initial (thermal) strain loading !29
THERMAL = ALPHA * DELTA_T ! strain !30
C = B (1, :) * M_E * THERMAL * AREA * BAR_L ! force !31
! ..... !32
! Weight load, in positive X-direction (wt density * volume) !33
C = C + (/ 0.5d0, 0.5d0 /) * GAMMA * AREA * BAR_L ! weight !34
! Save for stress post-processing (set post_el in keywords) !35
IF ( N_FILE1 > 0 ) WRITE (N_FILE1) M_E, B, THERMAL !36
! End of application dependent code !37

```

Figure 7.5.2 Implementation of an elastic linear bar

Interface from MODEL to POST_PROCESS_ELEM, 1			
Type	Status	Name	Remarks (keyword)
INTEGER (IN)		IE	Current element number
INTEGER (IN)		LT_FREE	Number of element type unknowns
INTEGER (IN)		LT_N	Number of element type solution nodes
INTEGER (IN)		N_R_B	Number of rows in B and E arrays (b_rows)
INTEGER (IN)		N_SPACE	Physical space dimension of problem (space)
INTEGER (IN)		N_FILE1	Optional user sequential unit (post_el)
INTEGER (IN)		N_FILE2	Optional user sequential unit (post_2)
REAL(DP) (IN)		D (LT_FREE)	Gathered element dof
REAL(DP) (OUT)		B (N_R_B, LT_FREE)	Gradient-dof transformation
REAL(DP) (OUT)		DGH (N_SPACE, LT_N)	Physical derivatives of H
REAL(DP) (OUT)		E (N_R_B, N_R_B)	Constitutive array
REAL(DP) (OUT)		H_INTG (LT_N)	Integral of H array
REAL(DP) automatic		XYZ (N_SPACE)	Coordinates of a point
REAL(DP) automatic		STRAIN_0 (N_R_B)	Initial strains
REAL(DP) automatic		STRAIN (N_R_B + 2)	Mechanical strains
REAL(DP) automatic		STRESS (N_R_B + 2)	Mechanical stresses

Figure 7.5.3 User interface to POST_PROCESS_ELEM, part 1

```

! ..... ! 1
! *** POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS FOLLOW *** ! 2
! ..... ! 3
! Define any new array or variable types, then give statements ! 4
! ..... ! 5
! AN AXIAL BAR BY DIRECT ENERGY APPROACH ! 6
! ELEMENT REAL PROPERTIES: (1) = AREA, (2) = ELASTIC MODULUS ! 7
! (3) = TEMP RISE, (4) = COEFF EXPANSION, (5) = WEIGHT DENSITY ! 8
! ..... ! 9
! STRESS = M_E * (MECHANICAL STRAIN - INITIAL STRAIN) !10
! ..... !11
REAL(DP) :: THERMAL, M_E ! initial strain, modulus !12
LOGICAL, SAVE :: FIRST = .TRUE. ! printing !13
! ..... !14
IF ( FIRST ) THEN ! first call !15
  FIRST = .FALSE. ; WRITE (6, 5) ! print headings !16
  5 FORMAT ( ' ELEMENT STRESSSES', /, & !17
    & ' ELEMENT STRESS MECH. STRAIN THERMAL STRAIN' ) !18
END IF ! first call !19
! ..... !20
!--> Read stress strain data from N_FILE1 (set by post_el) !21
READ (N_FILE1) M_E, B, STRAIN_0 (1) ! THERMAL = STRAIN_0 !22
! ..... !23
!--> Calculate mechanical strain, STRAIN = B * D !24
STRAIN (1) = DOT_PRODUCT ( B(1, :), D ) !25
! ..... !26
!--> Generalized Hooke's Law !27
STRESS (1) = M_E * (STRAIN (1) - STRAIN_0 (1)) !28
! ..... !29
WRITE (6, 1) IE, STRESS (1), STRAIN (1), STRAIN_0 (1) !30
1 FORMAT (I5, 3E15.5) !31
! *** END POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS *** !32

```

Figure 7.5.4 Stress recovery for the elastic bar

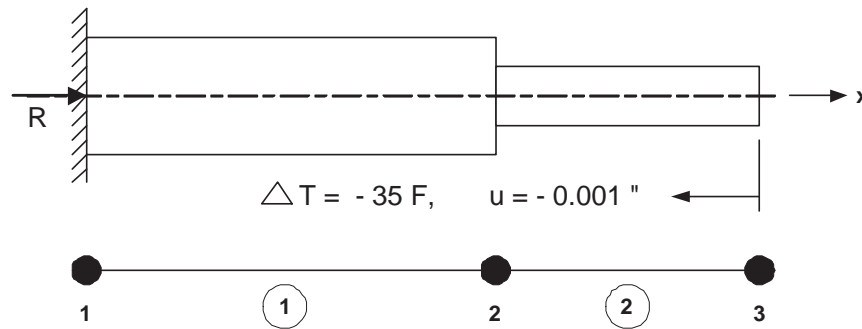


Figure 7.5.5 A thermally loaded elastic bar

summarize how they are obtained in the next section.

7.6 Reaction Flux Recovery for an Element

Regardless of whether we use variational methods or weighted residual methods we are often interested in post-processing to get the *flux recovery* data for some or all of the elements in the system. Once the assembled system has been solved for the primary nodal unknowns ϕ , we are often interested in also computing the nodal forces (or fluxes) that act on each individual element. For linear structural equilibrium, or thermal equilibrium, or a general Galerkin statement the algebraic equations are of the form

$$\mathbf{S}\mathbf{D} = \mathbf{C} + \mathbf{P} \quad (7.26)$$

where the square matrix \mathbf{S} is the assembly of the element square matrices \mathbf{S}^e and the column matrix \mathbf{C} is the sum of the consistent element force (or flux) matrices, \mathbf{C}^e , due to spatially distributed forces (or fluxes). Finally, \mathbf{P} is the vector of externally applied concentrated point forces (or fluxes) that are often called element reactions. The vector \mathbf{P} can also be thought of as an assembly of point sources on the elements, \mathbf{P}^e . This is always done if one is employing an element wavefront equation solving system. Most of the \mathbf{P}^e are identically zero. When P_j is applied to the j -th node of the system we simply find the element, e , where that node makes its first appearance in the data. Then, P_j is inserted in \mathbf{P}^e for that element and no entries are made in any other elements. If degree of freedom ϕ_j is given then P_j is an unknown reaction. To recover the concentrated "external" nodal forces or fluxes associated with a specific element we make the *assumption* that a similar expression holds for the element. That is,

$$\mathbf{S}^e \mathbf{D}^e = \mathbf{C}^e + \mathbf{P}^e \quad (7.28)$$

This is clearly exact if the system has only one element. Otherwise, it is a reasonable approximation. When we use an energy method to require equilibrium of an assembled system, we do not exactly enforce equilibrium in every element that makes up that system. Solving the reasonable approximation gives

$$\mathbf{P}^e = \mathbf{S}^e \mathbf{D}^e - \mathbf{C}^e \quad (7.29)$$

```

title "Steel-Brass cooled and deformed" ! begin keywords ! 1
nodes 3 ! Number of nodes in the mesh ! 2
elems 2 ! Number of elements in the system ! 3
dof 1 ! Number of unknowns per node ! 4
el_nodes 2 ! Maximum number of nodes per element ! 5
space 1 ! Solution space dimension ! 6
b_rows 1 ! Number of rows in the B (operator) matrix ! 7
shape 1 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex ! 8
remarks 2 ! Number of user remarks ! 9
el_real 5 ! Number of real properties per element !10
el_react ! Compute & list element reactions !11
post_el ! Require post-processing, create n_file1 !12
quit ! keyword input, remarks follow !13
Nodal displacements are exact, stresses too !14
Properties: A, E, DT, ALPHA, GAMMA (no gravity) !15
1 1 0.00 ! begin nodes !16
2 0 420. !17
3 1 660. !18
1 1 2 ! begin elements !19
2 2 3 !20
1 1 0. ! essential bc !21
3 1 -0.001 ! essential bc !22
1 10. 30.e6 -35. 6.7e-6 0. ! el, A, E, DT, ALPHA, GAMMA !23
2 8. 13.e6 -35. 12.5e-6 0. ! el, A, E, DT, ALPHA, GAMMA !24

```

Figure 7.5.6 Data for a thermally loaded deformed bar

```

*** INPUT SOURCE RESULTANTS *** ! 1
ITEM SUM POSITIVE NEGATIVE ! 2
1 0.0000E+00 7.0350E+04 -7.0350E+04 ! 3
! 4
*** REACTION RECOVERY *** ! 5
NODE, PARAMETER, REACTION, EQUATION ! 6
1, DOF_1, -5.4613E+04 1 ! 7
3, DOF_1, 5.4613E+04 3 ! 8
! 9
*** OUTPUT OF RESULTS IN NODAL ORDER *** !10
NODE, X-Coord, DOF_1, !11
1 0.0000E+00 0.0000E+00 !12
2 4.2000E+02 -2.2031E-02 !13
3 6.6000E+02 -1.0000E-03 !14
!15
** ELEMENT REACTION, INTERNAL SOURCES AND SUMMATIONS ** !16
ELEMENT 1 !17
NODE DOF REACTION ELEM_SOURCE SUMS !18
1 1 -5.46135E+04 7.03500E+04 !19
2 1 5.46135E+04 -7.03500E+04 !20
SUM: 1 0.00000E+00 0.00000E+00 0.00000E+00 !21
ELEMENT 2 !22
NODE DOF REACTION ELEM_SOURCE SUMS !23
2 1 -5.46135E+04 4.55000E+04 !24
3 1 5.46135E+04 -4.55000E+04 !25
SUM: 1 0.00000E+00 0.00000E+00 0.00000E+00 !26
!27
ELEMENT STRESSES !28
ELEMENT STRESS MECH. STRAIN THERMAL STRAIN !29
1 5.46135E+03 -5.24550E-05 -2.34500E-04 !30
2 6.82669E+03 8.76297E-05 -4.37500E-04 !31

```

Figure 7.5.7 Results for a thermally loaded deformed bar

where everything on the right hand side is known, since $\mathbf{D}^e \subset \mathbf{D}$ can be recovered as a gather operation. To illustrate these calculations consider the one-dimensional stepped Steel-Brass bar system given above in Fig. 7.4.2 where

$$\mathbf{S} = \frac{E^e A^e}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{C}^e = \frac{\mathbf{X}^e A^e L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

Now that all of the \mathbf{D} are known the \mathbf{D}^e can be extracted and substituted into Eq. 7.29 for each of the elements. For the first element Eq. 7.29 gives

$$\begin{aligned} \mathbf{P}^e &= 7.143 \times 10^5 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} 0 \\ 1.5638 \times 10^{-2} \end{Bmatrix} - \begin{Bmatrix} 594.3 \\ 594.3 \end{Bmatrix} \\ &= \begin{Bmatrix} -11,170.3 \\ 11,170.3 \end{Bmatrix} - \begin{Bmatrix} 594.3 \\ 594.3 \end{Bmatrix} = \begin{Bmatrix} -11,764.6 \\ 10,576.0 \end{Bmatrix} \text{ lb.} \end{aligned} \quad (7.30)$$

Likewise for the second element

$$\begin{aligned} \mathbf{P}^e &= 4.333 \times 10^5 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} 1.5638 \times 10^{-2} \\ 3.9381 \times 10^{-2} \end{Bmatrix} - \begin{Bmatrix} 288.0 \\ 288.0 \end{Bmatrix} \\ &= \begin{Bmatrix} -10,576.0 \\ 10,000.0 \end{Bmatrix} \text{ lb.} \end{aligned} \quad (7.31)$$

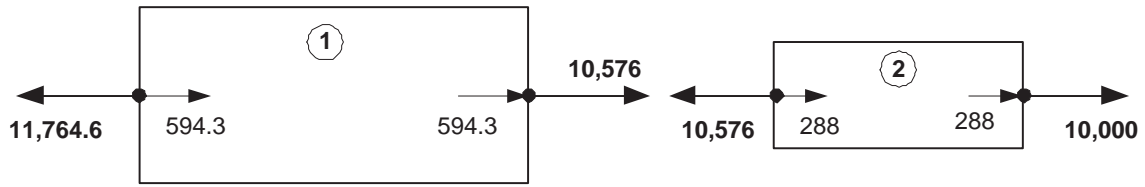
Note that if we choose to assemble these element \mathbf{P}^e values we obtain the system reactions \mathbf{P} . That is because element contributions at all unloaded nodes are equal and opposite (Newton's Third Law) and cancel when assembled. Figure 7.6.1. shows these "external" element forces when viewed on each element, as well as their assembly which matches the original system. This series of matrix operations is available in MODEL and is turned on only when the keyword *el_react* is present in the control data.

If we do the same reaction recovery for the second case of a thermal load and an enforced end displacement we get the values in Fig. 7.6.2. There is a subtle difference between these two cases. In the first case the gravity load creates a net external force source. In the second case the thermal loading creates equal and opposite internal loads that cancel for no net external source. That difference can be noted in the output listings of Figs. 7.4.4 and 6 where the "SUM" row of the "ELEM_SOURCE" column is non-zero in the first (gravity) case, but zero in the second (thermal) case.

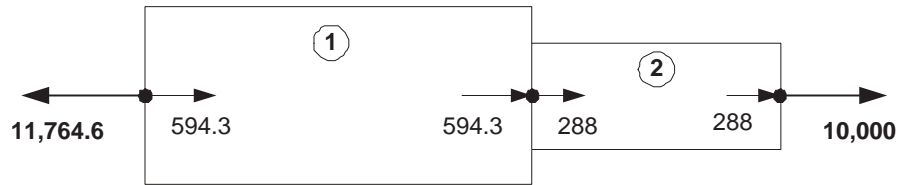
The reader is warned to remember that these calculations in Eqs. 7.29 have been carried out in the global coordinate systems. In more advanced structural applications it is often desirable to transform the \mathbf{P}^e back to element local coordinate system. For example, with a general truss member we are more interested in the force along the line of action of the bar rather than its x and y components. Sometimes we list both results and the user selects which is most useful. The necessary coordinate transformation is of the form

$$\mathbf{P}_L^e = \mathbf{T}^e \mathbf{P}_g^e \quad (7.32)$$

where the square rotation matrix, \mathbf{T} , contains the direction cosines between the global

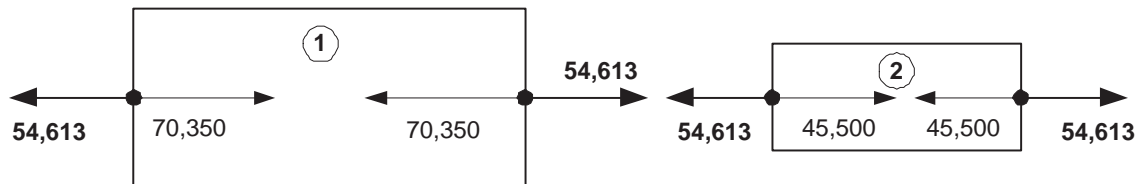


a) Element level reactions and loads

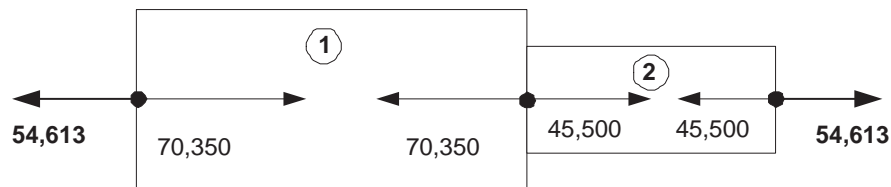


b) System reactions and loads

Figure 7.6.1 Element equilibrium reaction recovery, case 1



a) Element level reactions and loads



b) System reactions and loads

Figure 7.6.2 Element equilibrium reaction recovery, case 2

axis, g , and the element local axis, L .

7.7 Heat Transfer in a Rod

A problem closely related to the previous problem is that of steady state heat transfer. Consider the heat transfer in a slender rod that has a specified temperature, θ_0 , at $x = 0$ and is insulated at the other end, $x = L$. The rod has cross-sectional area, A , with a thermal conductivity of K . Thus, the rod conducts heat along its length. The rod is also surrounded by a convecting medium with a uniform temperature of θ_∞ . Thus, the

rod also convects heat on its outer surface area. Let the convective transfer coefficient be h and the outer perimeter of the rod be P . The governing differential equation for the temperature, $\theta(x)$, is given by Myers [6] as

$$KA \frac{d^2\theta}{dx^2} - hP(\theta - \theta_\infty) = 0, \quad 0 < x < L \quad (7.33)$$

with the essential condition $\theta(0) = \theta_0$ and the natural boundary condition $d\theta/dx(L) = 0$, which corresponds to an insulated right end. The exact solution can be shown to be $\theta(x) = \theta_\infty + (\theta_0 - \theta_\infty) \cosh[m(L-x)]/\cosh(mL)$ where $m^2 = hP/KA$ is a non-dimensional measure of the relative importance of convection (hP) and conduction (KA). This problem can be identified as the Euler equation of a variational principle. This principle will lead to system equations that are structured differently from our previous example with the bar. In that case, the boundary integral contributions (tractions) defined a column matrix and thus went on the right hand side of the system equations. Here we will see that the boundary contributions (convection) will also define a square matrix. Thus, they will go into the system coefficients on the left hand side of the system equations.

Generally a variational formulation of steady state heat transfer involves volume integrals containing conduction terms and surface integrals with boundary heat flux, e.g., convection, terms. In our one-dimensional example both the volume and surface definitions involve an integral along the length of the rod. Thus, the distinction between volume and surface terms is less clear and the governing functional given by Myers is simply stated as a line integral. Specifically, one must render stationary the functional

$$I(\theta) = \frac{1}{2} \int_0^L [K A (d\theta/dx)^2 + h P \theta^2] dx \quad (7.34)$$

$$- \int_0^L h P \theta \theta_\infty dx + q_0 \theta(0) - q_L \theta(L)$$

subject to the essential boundary condition(s). Divide the rod into a number of nodes and elements and introduce a finite element model where we assume

$$I = \sum_{e=1}^{n_e} I^e + \sum_{b=1}^{n_b} I^b + I^r$$

where we have defined a typical element volume contribution of

$$I^e = \frac{1}{2} \int_{L^e} K^e A^e (d\theta^e/dx)^2 dx \quad (7.35)$$

and typical boundary contribution is

$$I^b = \frac{1}{2} \int_{L^b} h^b P^b \theta^{b2} dx - \int_{L^b} h^b P^b \theta^b \theta_\infty dx, \quad (7.36)$$

and I^r denotes any point flux sources or non-zero reaction contributions (here q_0 and/or q_L) that may be present. Most authors move known point sources or sinks into I^b and leave only the one or two unknown reaction terms in I^r . Using our interpolation relations as before $\theta^e(x) = \mathbf{H}^e(x) \mathbf{D}^e = \mathbf{D}^{eT} \mathbf{H}^{eT}$, and again in this special case $\theta^b(x) = \theta^e(x)$. Thus, these can be written symbolically as $I^e = \frac{1}{2} \mathbf{D}^{eT} \mathbf{S}^e \mathbf{D}^e$, $I^b = \frac{1}{2} \mathbf{D}^{bT} \mathbf{S}^b \mathbf{D}^b - \mathbf{D}^{bT} \mathbf{C}^b$.

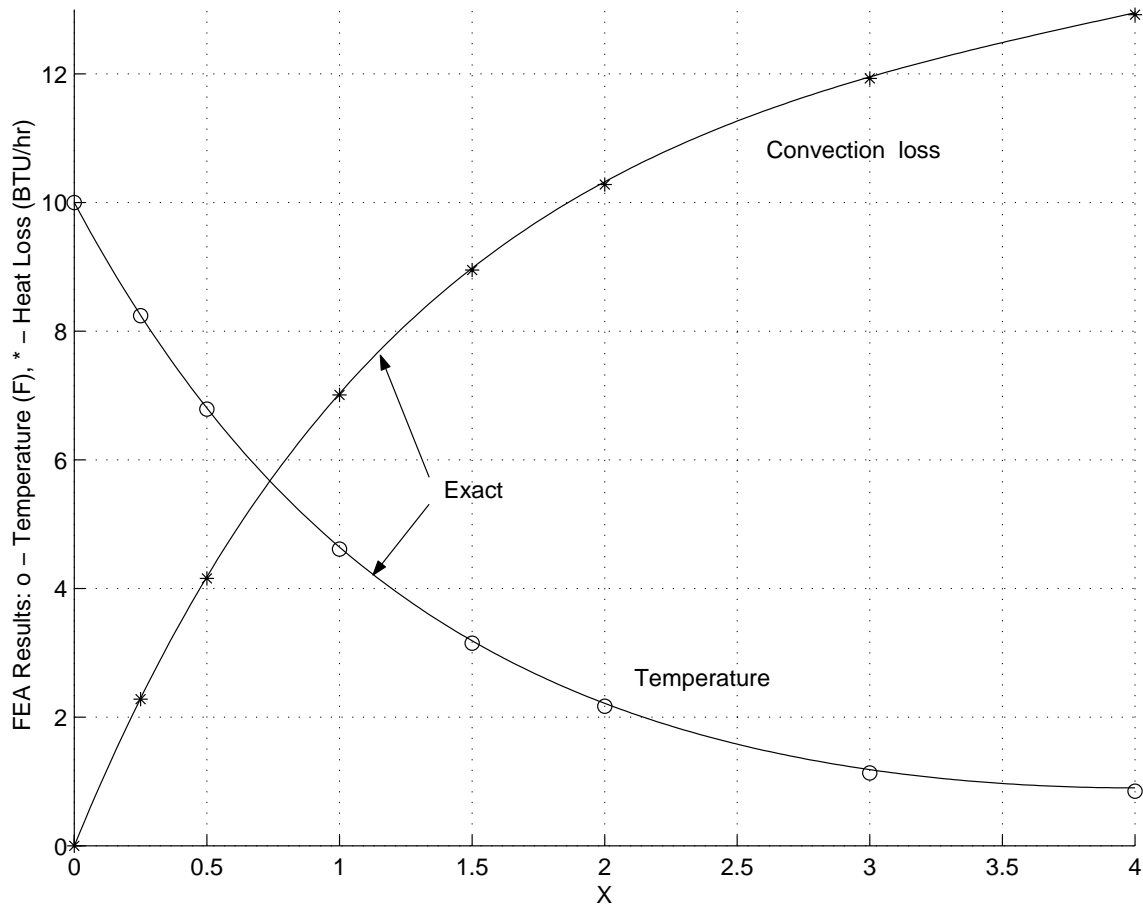
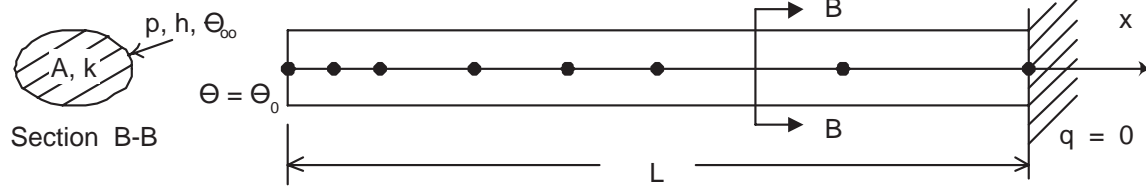


Figure 7.7.1 Temperatures (o) and convection losses (*) in a slender rod

Assuming constant properties and the linear interpolation in Eq. 7.24 the element and boundary matrices reduce to

$$\mathbf{S}^e = \frac{K^e A^e}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{S}^b = \frac{h^b P^b L^b}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{C}^b = \frac{\theta_\infty h^b P^b L^b}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

Note that the conduction effect is inversely proportional to the material length while the convection effect is directly proportional to the length. If the problem is normalized so $\theta_\infty = 0$ then there is no column matrix defined and the equations will be homogeneous. As before, the assembled system equations are $\mathbf{S}\mathbf{T} = \mathbf{C}$ where \mathbf{S} is the direct assembly of \mathbf{S}^e and \mathbf{S}^b and \mathbf{C} is assembled from the \mathbf{C}^b .

Another aspect of interest here is how to *post-process* the results so as to determine the convective heat loss. It should be equal and opposite to the sum of any external heat flux reactions necessary to maintain essential boundary conditions. The convection heat loss, at any point, is $dq = hP(\theta - \theta_\infty) dx$ where $(\theta - \theta_\infty)$ is the surface temperature difference. On a typical boundary segment this simplifies (for constant boundary data) to

$$Q^b = \int_{L^b} dq = \int_{L^b} h^b P^b [\theta^b(x) - \theta_\infty] dx = h^b P^b \int_{L^b} \mathbf{H}^b(x) dx [\mathbf{D}^b - \boldsymbol{\theta}_\infty] \quad (7.37)$$

where $\boldsymbol{\theta}_\infty$ is a vector with the boundary nodal values of θ_∞ . For a linear element interpolation, as above, it is

$$Q^b = \frac{1}{2} h^b P^b L^b [1 \quad 1] (\mathbf{D}^b - \boldsymbol{\theta}_\infty) = \mathbf{P}^b (\mathbf{D}^b - \boldsymbol{\theta}_\infty). \quad (7.38)$$

Thus, if the constant array \mathbf{P}^b is computed and stored for each segment then once all the temperatures are computed the boundary sub-set \mathbf{D}^b can be gathered along with \mathbf{P}^b to compute the loss Q^b . This is the first of several applications where we see that sometimes the post-processing will require the spatial integral of the solution and not just its gradient. Summing on the total boundary (all elements in this special case) gives the total heat loss. That value would, of course, equal the heat entering at the end $x = 0$. As a specific numerical example let $L = 4$ ft., $A = 0.01389$ ft², $h = 2$ BTU/hr – ft² F, $K = 120$ BTU/hr – ft F, $P = 0.5$ ft., and $t_0 = 10$ F. The mesh selected for this analysis are shown in Fig. 7.7.1 along with the results of the finite element analysis. A general implementation of this model via numerical integration is given in Fig. 7.7.2. It is valid for any member of the element library (currently linear through cubic interpolation). The input data for the application are given in Fig. 7.7.3 and selected corresponding output sets are in Fig. 7.7.4. The typical coding for recovering the integral of the product of the constant convection data and the interpolation function is shown in Fig. 7.7.5.

7.8 Element Validation *

The successful application of finite element analysis should always include a validation of the element to be used and its implementation in a specific computer program. Usually, the elements utilized in most problem classes are very well understood and tested. However, some applications can be difficult to model, and the elements used for their analysis may be more prone to numerical difficulties. Therefore, one should subject elements to be used to a series of element validation tests. Two of the most common and important tests are the *patch test* introduced by Irons [3–5] and the *single-element tests* proposed by Robinson [8]. The single-element tests generally show the effects of element geometrical parameters such as convexity, aspect ratio, skewness, taper, and out-of-plane warpage. It is most commonly utilized to test for a sensitivity to element aspect ratio. The single-element test usually consists of taking a single element in rectangular, triangular, or line form, considering it as a complete domain, and then investigating its behavior for various load or boundary conditions as a geometrical parameter is varied. An analytical solution is usually available for such a test.

The patch test has been proven to be a valid convergence test. It was developed from physical intuition and later written in mathematical forms. The basic concept is

```

! ..... ! 1
!   ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW *** ! 2
! ..... ! 3
! Combined heat conduction through, convection from a bar: ! 4
!   K*A*U,XX - h*P*(U-U_ext) = 0, U(0)=U_0, dU/dx(L)=0 ! 5
! For globally constant data the analytic solution is: ! 6
! U(x) = U_ext - (U_0-U_ext) * cosh [m*(L-x)]/ cosh [mL] ! 7
! where m^2 = h_e*P_e/(K_e*A_e), dimensionless. ! 8
! Real element properties are: ! 9
! 1) K_e = conductivity, BTU/ hr ft F !10
! 2) A_e = area of bar, ft^2 !11
! 3) h_e = convection, BTU/ hr ft^2 F !12
! 4) P_e = perimeter of area A_e, ft !13
! Miscellaneous real FE data: !14
! 1) U_ext = external reference temperature, F !15
! Miscellaneous real data used ONLY for analytic solution: !16
! 2) L = exact length, ft !17
! 3) U_0 = essential bc at x = 0, F !18
!19
REAL(DP) :: DL, DX_DR ! Length, Jacobian !20
REAL(DP) :: K_e, A_e, h_e, P_e, U_ext ! properties !21
INTEGER :: IQ ! Loops !22
!23
DL = COORD (LT_N, 1) - COORD (1, 1) ! LENGTH !24
DX_DR = DL / 2. ! CONSTANT JACOBIAN !25
!26
U_ext = GET_REAL_MISC (1) ! external temperature !27
K_e = GET_REAL_LP (1) ! thermal conductivity !28
A_e = GET_REAL_LP (2) ! area of bar !29
h_e = GET_REAL_LP (3) ! convection coefficient on perimeter !30
P_e = GET_REAL_LP (4) ! perimeter of area A_e !31
!32
E = K_e * A_e ! constitutive array !33
!34
CALL STORE_FLUX_POINT_COUNT ! Save LT_QP, for post & error !35
! S, C, H_INTG already zeroed !36
DO IQ = 1, LT_QP ! LOOP OVER QUADRATURES !37
!38
! GET INTERPOLATION FUNCTIONS, AND X-COORD !39
H = GET_H_AT_QP (IQ) !40
XYZ = MATMUL (H, COORD) ! ISOPARAMETRIC !41
!42
! LOCAL AND GLOBAL DERIVATIVES !43
DLH = GET_DLH_AT_QP (IQ) ! local !44
DGH = DLH / DX_DR ! global !45
!46
! CONVECTION SOURCE !47
C = C + h_e * P_e * U_ext * H * WT (IQ) * DX_DR !48
!49
! SQUARE MATRIX, CONDUCTION & CONVECTION !50
S = S + ( K_e * A_e * MATMUL (TRANPOSE(DGH), DGH) & !51
+ h_e * P_e * OUTER_PRODUCT (H, H) ) * WT (IQ) * DX_DR !52
!53
! INTEGRATING FOR CONVECTION LOSS, FOR POST PROCESSING !54
H_INTG = H_INTG + h_e * P_e * H * WT (IQ) * DX_DR !55
!56
! SAVE FOR FLUX AVERAGING OR POST PROCESSING, B == DGH !57
CALL STORE_FLUX_POINT_DATA (XYZ, E, DGH) ! for error est too !58
END DO ! QUADRATURE !59
IF ( N_FILE1 > 0) WRITE (N_FILE1) H_INTG, U_ext ! if "post_el" !60

```

Figure 7.7.2 A numerically integrated thermal element

```

title "Myer's 1-D heat transfer example, 7 L2 " ! keywords ! 1
exact_case 1 ! Analytic solution for list_exact, etc ! 2
list_exact ! List given exact answers at nodes, etc ! 3
nodes 8 ! Number of nodes in the mesh ! 4
elems 7 ! Number of elements in the system ! 5
dof 1 ! Number of unknowns per node ! 6
el_nodes 2 ! Maximum number of nodes per element ! 7
space 1 ! Solution space dimension ! 8
b_rows 1 ! Number of rows in the B (operator) matrix ! 9
shape 1 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex !10
el_react ! Compute & list element reactions !11
remarks 21 ! Number of user remarks !12
gauss 2 ! Maximum number of quadrature point !13
el_real 4 ! Number of real properties per element !14
reals 3 ! Number of miscellaneous real properties !15
el_homo ! Element properties are homogeneous !16
post_el ! Require post-processing, create n_file1 !17
no_error_est ! Do NOT compute SCP element error estimates !18
quit ! keyword input, remarks follow !19
Combined heat conduction through, convection from a bar: !20
K*A*U,XX - h*P*(U-U_ext) = 0, U(0)=U_0, dU/dx(L)=0 !21
For globally constant data the analytic solution is: !22
U(x) = U_ext - (U_0-U_ext) * cosh [m*(L-x)] / cosh [mL] !23
where m^2 = h_e*P_e/(K_e*A_e), dimensionless. !24
Real element properties are: !25
1) K_e = conductivity, BTU/ hr ft F !26
2) A_e = area of bar, ft^2 !27
3) h_e = convection, BTU/ hr ft^2 F !28
4) P_e = perimeter of area A_e, ft !29
Miscellaneous real FE data: !30
1) U_ext = external reference temperature, F !31
Miscellaneous real data used ONLY for analytic solution: !32
2) L = exact length, ft !33
3) U_0 = essential bc at x = 0, F !34
Element convection loss is difference in elem reactions !35
e.g.: e=1 node 1 node 2 !36
12.92 BTU ---->*_____(1)_____* ----> 10.64 BTU !37
Elem convection loss: 2.28 BTU ----> (physical check) !38
Note: System reaction, 12.92 BTU (12.86 exact), offsets the !39
sum of the element convection losses, 12.92 BTU (phys chk) !40
1 1 0.00 ! begin nodes: node, bc flag, x !41
2 0 0.25 !42
3 0 0.50 !43
4 0 1.00 !44
5 0 1.50 !45
6 0 2.00 !46
7 0 3.00 !47
8 0 4.00 !48
1 1 2 ! begin elements !49
2 2 3 !50
3 3 4 !51
4 4 5 !52
5 5 6 !53
6 6 7 !54
7 7 8 !55
1 1 10. ! essential bc: node, dof, value !56
1 120. 0.01389 2.0 0.5 ! el, K_e A_e h_e P_e (homogeneous) !57
0.0 4.0 10.0 ! Miscellaneous: U_ext L U_0 !58

```

Figure 7.7.3 Example convection data

```

*** REACTION RECOVERY *** ! 1
NODE, PARAMETER, REACTION, EQUATION ! 2
1, DOF_1, 1.2921E+01 1 ! 3
! 4
*** OUTPUT OF RESULTS AND EXACT VALUES IN NODAL ORDER *** ! 5
NODE, X-Coord, DOF_1, EXACT1, ! 6
1 0.0000E+00 1.0000E+01 1.0000E+01 ! 7
2 2.5000E-01 8.2385E+00 8.2475E+00 ! 8
3 5.0000E-01 6.7878E+00 6.8051E+00 ! 9
4 1.0000E+00 4.6138E+00 4.6438E+00 !10
5 1.5000E+00 3.1496E+00 3.1877E+00 !11
6 2.0000E+00 2.1700E+00 2.2157E+00 !12
7 3.0000E+00 1.1322E+00 1.1847E+00 !13
8 4.0000E+00 8.4916E-01 9.0072E-01 !14
!15
** ELEMENT REACTION, INTERNAL SOURCES AND SUMMATIONS ** !16
ELEMENT 1 !17
NODE DOF REACTION ELEM_SOURCE SUMS !18
1 1 1.29210E+01 0.00000E+00 !19
2 1 -1.06412E+01 0.00000E+00 !20
SUM: 1 2.27981E+00 0.00000E+00 2.27981E+00 Note !21
ELEMENT 2 !22
NODE DOF REACTION ELEM_SOURCE SUMS !23
2 1 1.06412E+01 0.00000E+00 !24
3 1 -8.76294E+00 0.00000E+00 !25
SUM: 1 1.87829E+00 0.00000E+00 1.87829E+00 !26
ELEMENT 3 !27
NODE DOF REACTION ELEM_SOURCE SUMS !28
3 1 8.76294E+00 0.00000E+00 !29
4 1 -5.91252E+00 0.00000E+00 !30
SUM: 1 2.85041E+00 0.00000E+00 2.85041E+00 !31
ELEMENT 4 !32
NODE DOF REACTION ELEM_SOURCE SUMS !33
4 1 5.91252E+00 0.00000E+00 !34
5 1 -3.97165E+00 0.00000E+00 !35
SUM: 1 1.94087E+00 0.00000E+00 1.94087E+00 !36
ELEMENT 5 !37
NODE DOF REACTION ELEM_SOURCE SUMS !38
5 1 3.97165E+00 0.00000E+00 !39
6 1 -2.64175E+00 0.00000E+00 !40
SUM: 1 1.32990E+00 0.00000E+00 1.32990E+00 !41
ELEMENT 6 !42
NODE DOF REACTION ELEM_SOURCE SUMS !43
6 1 2.64175E+00 0.00000E+00 !44
7 1 -9.90679E-01 0.00000E+00 !45
SUM: 1 1.65107E+00 0.00000E+00 1.65107E+00 !46
ELEMENT 7 !47
NODE DOF REACTION ELEM_SOURCE SUMS !48
7 1 9.90679E-01 0.00000E+00 !49
8 1 0.00000E+00 0.00000E+00 !50
SUM: 1 9.90679E-01 0.00000E+00 9.90679E-01 !51
** ELEMENT CONVECTION HEAT LOSS ** !52
1 2.27980 <- Note reaction above, etc. !53
2 1.87828 !54
3 2.85041 !55
4 1.94087 !56
5 1.32989 !57
6 1.65107 !58
7 0.99067 !59
TOTAL HEAT LOSS = 12.92103 (Exact = 12.858) !60

```

Figure 7.7.4 Selected conduction-convection output

```

! ..... ! 1
! *** POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS FOLLOW *** ! 2
! ..... ! 3
! Define any new array or variable types, then give statements. ! 4
! H_INTG (LT_N) Integral of interpolation functions, H, available ! 5
! ..... ! 6
! Linear line element face convection heat loss recover ! 7
REAL(DP) :: U_ext ! external temperature ! 8
REAL(DP), SAVE :: Q_LOSS, TOTAL ! Face and total heat loss ! 9
LOGICAL, SAVE :: FIRST = .TRUE. ! printing !10
! ..... !11
IF ( FIRST ) THEN ! first call !12
  FIRST = .FALSE. ; WRITE (6, 5) ! print headings !13
  5 FORMAT ('*** CONVECTION HEAT LOSS ***', /, & !14
    & 'ELEMENT HEAT_LOST') !15
  TOTAL = 0.d0 ! initialize !16
END IF ! first call !17
! ..... !18
! Get previously integrated interpolation function, times !19
! the convection properties, h_e * P_e, now stored in H_INTG; !20
! and the surrounding gas temperature, U_ext, that were !21
! saved in ELEM_SQ_MATRIX. (Indicated by keyword post_el.) !22
! U_ext = GET_REAL_MISC (1) ! external temperature !23
! h_e = GET_REAL_LP (3) ! perimeter convection coefficient !24
! P_e = GET_REAL_LP (4) ! perimeter length of area !25
! ..... !26
IF ( N_FILE1 > 0 ) READ (N_FILE1) H_INTG, U_ext ! if "post_el" !27
! ..... !28
! HEAT LOST : Integral over bar length of hp * (T - T_inf) !29
D (1:LT_N) = D(1:LT_N) - U_ext ! Temp difference at nodes !30
Q_LOSS = DOT_PRODUCT (H_INTG, D) ! Face loss integral !31
TOTAL = TOTAL + Q_LOSS ! Running total !32
! ..... !33
PRINT '(I6, ES15.5)', IE, Q_LOSS !34
IF ( IE == N_ELEMS ) PRINT *, 'TOTAL = ', TOTAL !35
! *** END POST_PROCESS_ELEM PROBLEM DEPENDENT STATEMENTS *** !36

```

Figure 7.7.5 Element convection heat loss recovery

fairly simple. Imagine what happens as one introduces a very large, almost infinite, number of elements. Clearly, they would become very small in size. If we think of the quantities being integrated to form the element matrices, we can make an observation about how the solution would behave in this limit. The integrand, such as the strain energy, contains derivative terms that would become constant as the element size shrinks toward zero. Thus, to be valid in the limit, the element formulation must, at least, be able to yield the correct results in that state. That is, to be assured of convergence one must be able to exactly satisfy the state where the derivatives, in the governing integral statement, take on constant or zero values. This condition can be stated as a mathematical test or as a simple numerical test. The latter option is what we want here. The patch test provides a simple numerical way for a user to test an element, or complete computer program, to verify that it behaves as it should.

We define a patch of elements to be a mesh where at least one node is completely surrounded by elements. Any node of this type is referred to as an *interior node*. The other nodes are referred to as *exterior* or *perimeter nodes*. We will compute the dependent variable at all interior nodes. The derivatives of the dependent variable will be

computed in each element. The perimeter nodes are utilized to introduce the essential boundary conditions and/or loads required by the test. Assume that the governing integral statement has derivatives of order n . We would like to find boundary conditions that would make those derivatives constant. This can be done by selecting an arbitrary n -th order polynomial function of the global coordinates to describe the dependent variable in the global space that is covered by the patch mesh. Clearly, the n -th order derivatives of such a function would be constant as desired. The assumed polynomial is used to define the essential boundary conditions on the perimeter nodes of the patch mesh. This is done by substituting the input coordinates at the perimeter nodes into the assumed function and computing the required value of the dependent variable at each such node. Once all of the perimeter boundary conditions are known, the solution can be numerically executed. The resulting values of the dependent variable are computed at each interior node. To pass the patch test, these computed internal values must agree with the value found when their internal nodal coordinates are substituted into the assumed global polynomial. However, the real test is that when each element is checked, the calculated n -th order derivatives must agree with the arbitrary assumed values used to generate the global function. If an element does not satisfy this test, it should not be used. The patch test can also be used for other purposes. For example, the analyst may wish to distort the element shape and/or change the numerical integration rule to see what effect that has on the numerical accuracy of the patch test.

As a simple elementary example of an analytic solution of the patch test, consider the bar element. The smallest possible patch is one with two line elements. Such a patch has two exterior nodes and one interior node. For simplicity, let the lengths of the two elements be equal and have a value of L . The governing integral statement contains only the first derivative of u . An arbitrary linear function can be selected for the patch test, since it would have a constant first derivative. Therefore, select $u(x) = a + bx$ for $0 \leq x \leq 2L$, where a and b are arbitrary constants. Assembling a two-element patch:

$$\frac{AE}{L} \begin{bmatrix} 1 & -1 & 0 \\ -1 & (1+1) & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} P_1 \\ 0 \\ P_3 \end{Bmatrix}$$

where P_1 and P_3 are the unknown reactions associated with the prescribed external displacements. These two exterior patch boundary conditions are obtained by substituting their nodal coordinates into the assumed patch solution :

$$u_1 = u(x_1) = a + b(0) = a, \quad u_3 = u(x_2) = a + b(2L) = a + 2bL.$$

Modifying the assembled equations to include the patch boundary conditions gives

$$\frac{AE}{L} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{Bmatrix} a \\ u_2 \\ a + 2bL \end{Bmatrix} = \begin{Bmatrix} P_1 \\ 0 \\ P_3 \end{Bmatrix} - \frac{aAE}{L} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} - \frac{(a + bL) AE}{L} \begin{Bmatrix} 0 \\ -1 \\ 1 \end{Bmatrix}.$$

Retaining the independent second equation gives the displacement relation

$$\frac{2AE}{L} u_2 = 0 + \frac{aAE}{L} + \frac{(a + 2bL) AE}{L}.$$

Thus, the internal patch displacement is $u_2 = (2a + 2bL)/2 = (a + bL)$. The value required by the patch test is $u(x_2) = (a + bx_2) = (a + bL)$. This agrees with the computed solution, as required by a valid element. The element strains are

$$e = 1 : \quad \varepsilon = \frac{(u_2 - u_1)}{L} = \frac{[(a + bL) - a]}{L} = b$$

$$e = 2 : \quad \varepsilon = \frac{(u_3 - u_2)}{L} = \frac{[(a + 2bL) - (a + bL)]}{L} = b.$$

Thus, all element derivatives are constant. However, these constants must agree with the constant assumed in the patch. That value is $\varepsilon = du/dx = d(a + bx)/dx = b$. Therefore, the patch test is completely satisfied. At times one also wishes to compute the reactions, i.e., P_1 and P_3 . To check for possible rank deficiency in the element formulation, one should repeat the test with only enough displacements prescribed to prevent rigid body motion. (That is, to render the square matrix non-singular.) Then, the other outer perimeter nodes are loaded with the reactions found in the precious patch test. In the above example, substituting u_1 and u_2 into the previously discarded first equation yields the reaction $P_1 = -bAE$. Likewise, the third equation gives $P_3 = -P_1$, as expected. Thus, the above test could be repeated by prescribing u_1 and P_3 , or P_1 and u_3 . The same results should be obtained in each case. A major advantage of the patch test is that it can be carried out numerically. In the above case, the constants a and b could have been assigned arbitrary numerical values. Inputting the required numerical values of A , E and L would give a complete numerical description that could be tested in a standard program. Such a procedure also verifies that the computer program satisfies certain minimum requirements. A problem with some elements is that they can pass the patch test for a uniform mesh, but fail when an arbitrary irregular mesh is employed. Thus, as a general rule, one should try to avoid conducting the test with a regular mesh, such as that given in the above example. It would have been wiser to use unequal element lengths such as L and αL , where α is an arbitrary constant. The linear bar element should pass the test for any scaling ratio, α . However, for α near zero, numerical ill-conditioning begins to affect the answers. Data and results for the patch test of a linear bar element are shown in Figs. 7.8.1 and 2.

7.9 Euler's Equations of Variational Calculus *

Euler's Theorem of Variational Calculus was given in Eqs. 1.6 - 8. When the value of ϕ is given on a portion of the boundary, Γ , we call that an essential boundary condition, or a Dirichlet type condition. When q is present in the boundary condition (of Eq. 1.8), but a is zero that case is called a Neumann type condition, or flux condition. The most common Neumann type is the 'natural condition', $q = 0$, since then the boundary integral in Eq. 1.6 does not have to be evaluated (it is satisfied naturally). The most general form of Eq. 1.8 is called a 'mixed boundary condition', which is also known as a 'Robin type' condition. Note that a Robin condition involves a non-zero a value in Eq. 1.6 and thus it will contribute a new term to the square matrix (since u^2 leads to a

```

title "Elastic linear bar patch test" ! begin keywords ! 1
nodes 3 ! Number of nodes in the mesh ! 2
elems 2 ! Number of elements in the system ! 3
dof 1 ! Number of unknowns per node ! 4
el_nodes 2 ! Maximum number of nodes per element ! 5
space 1 ! Solution space dimension ! 6
b_rows 1 ! Number of rows in the B (operator) matrix ! 7
shape 1 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex ! 8
remarks 3 ! Number of user remarks ! 9
el_real 5 ! Number of real properties per element !10
el_homo ! Element properties are homogeneous !11
el_list ! List results at each node of each element !12
el_react ! Compute & list element reactions !13
post_el ! Require post-processing, create n_file1 !14
quit ! keyword input, remarks follow !15
Assume u = a + bx. Let a = 5, b= -4 so that u(0) = u_1 = 5 !16
and u(2) = u_3 = -3 then all mechanical strains = -4 !17
Properties: A, E, DT, ALPHA, GAMMA (any homogeneous) !18
  1 1 0.0 ! node, bc flag, x !19
  2 0 1.0 ! answer here must match 'a + bx' !20
  3 1 2.0 ! node, bc flag, x !21
    1 1 2 ! begin elements !22
    2 2 3 !23
1 1 5. ! node, dof, essential bc: u(0) = u_1 = 5 !24
3 1 -3. ! node, dof, essential bc: u(2) = u_3 = -3 !25
1 1. 10. 0. 0. 0. ! el, A, E, DT, ALPHA, GAMMA (any A,E) !26

```

Figure 7.8.1 Example patch test data for a bar

```

*** REACTION RECOVERY *** ! 1
NODE, PARAMETER, REACTION, EQUATION ! 2
  1, DOF_1, 4.0000E+01 1 ! 3
  3, DOF_1, -4.0000E+01 3 ! 4
! 5
*** EXTREME VALUES OF THE NODAL PARAMETERS *** ! 6
PARAMETER MAXIMUM, NODE MINIMUM, NODE ! 7
DOF_1, 5.0000E+00, 1 -3.0000E+00, 3 ! 8
! 9
*** OUTPUT OF RESULTS IN NODAL ORDER *** !10
NODE, X-Coord, DOF_1, !11
  1 0.0000E+00 5.0000E+00 !12
  2 1.0000E+00 1.0000E+00 ! passes patch test !13
  3 2.0000E+00 -3.0000E+00 !14
!15
** ELEMENT REACTION, AND INTERNAL SOURCES ** !16
ELEMENT 1 !17
NODE DOF REACTION ELEM_SOURCE !18
  1 1 4.00000E+01 0.00000E+00 !19
  2 1 -4.00000E+01 0.00000E+00 !20
ELEMENT 2 !21
NODE DOF REACTION ELEM_SOURCE !22
  2 1 4.00000E+01 0.00000E+00 !23
  3 1 -4.00000E+01 0.00000E+00 !24
!25
ELEMENT STRESS ! pass !26
ELEMENT STRESS MECH. STRAIN THERMAL STRAIN !27
  1 -4.00000E+01 -4.00000E+00 0.00000E+00 !28
  2 -4.00000E+01 -4.00000E+00 0.00000E+00 !29

```

Figure 7.8.2 Patch test results for a bar

quadratic form in \mathbf{H}). Likewise, one could reduce this to a one-dimensional form

$$I = \int_a^b f\left(x, \phi, \frac{d\phi}{dx}\right) dx + (q + a\phi) \Big|_{x=b} - (q + a\phi) \Big|_{x=a} \quad (7.45)$$

so the ordinary differential equation (ODE) is

$$\frac{\partial f}{\partial \phi} - \frac{d}{dx} \frac{\partial f}{\partial (d\phi/dx)} = 0 \quad (7.46)$$

and the natural boundary condition if ϕ is not prescribed is

$$n_x \frac{\partial f}{\partial (d\phi/dx)} + q + a\phi = 0. \quad (7.47)$$

Fourth order equations are formulated in the same way. For example,

$$I(\phi) = \int_a^b f\left(x, \phi, \frac{d\phi}{dx}, \frac{d^2\phi}{dx^2}\right) dx, \quad (7.48)$$

a functional involving the second derivative of ϕ , has the Euler equation

$$\frac{\partial f}{\partial \phi} - \frac{d}{dx} \frac{\partial f}{\partial (d\phi/dx)} + \frac{d^2}{dx^2} \frac{\partial f}{\partial (d^2\phi/dx^2)} = 0, \quad (7.49)$$

and the natural boundary condition of

$$\frac{\partial f}{\partial (d\phi/dx)} - \frac{d}{dx} \frac{\partial f}{\partial (d^2\phi/dx^2)} = 0 \quad (7.50)$$

if $d\phi/dx$ is specified (and ϕ unknown), and the natural condition of

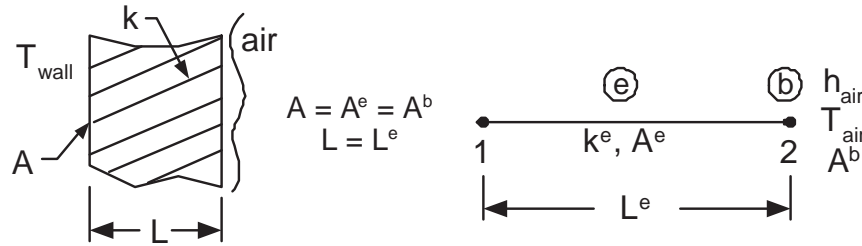
$$\frac{\partial f}{\partial (d^2\phi/dx^2)} = 0 \quad (7.51)$$

if ϕ is specified (and $d\phi/dx$ is unknown).

7.10 Exercises

1. Consider heat transfer through a planar wall with a given temperature on one side and convection on the other. Employ a single linear element model where ϕ_1 is the inside temperature at left node 1, and where right node ϕ_2 is the other surface temperature adjacent to the convection fluid. Heat is convected away there at the rate of $q_h = h_a A_a (\phi_2 - \theta_a)$, where h_a is the convection coefficient over the convection area of A_a which is adjacent to the convecting fluid at temperature θ_a . The constant conduction matrix was given earlier and, in general, the convection matrices on boundary segment b are $\mathbf{S}_h^b = \int_{\Gamma^b} \mathbf{H}^{bT} \mathbf{H}^b h^b d\Gamma$, and $\mathbf{C}_h^b = \int_{\Gamma^b} \mathbf{H}^{bT} h^b \phi_a^b d\Gamma$. At a point surface approximation (Γ^b) the \mathbf{H}^b is constant (unity) and the convection data (h_a^b, θ_a^b) are also constant. Create the (scalar) boundary terms, assemble, and solve for ϕ_2 . Verify the result is $\phi_2 = \frac{A_a h_a \theta_a + (k/L) \phi_1 A^e}{(k A^e / L + h A_a)}$. Simplify for the present case where the conducting and

convecting areas are the same, $A^e = A_a = A^b$. Consider the two special cases of $h = 0$ and $h = \infty$ and discuss what they then represent as boundary conditions (Dirichlet, or Neumann).



Problem P7.1 Wall conduction-convection

2. Consider a two-dimensional functional

$$I = \frac{1}{2} \int_{\Omega} \left[K_x \left(\frac{\partial u}{\partial x} \right)^2 + K_y \left(\frac{\partial u}{\partial y} \right)^2 - 2Qu \right] d\Omega + \int_{\Gamma} \left(qu + au^2/2 \right) d\Gamma.$$

Show that the partial differential equation from Euler's theorem is

$$-Q - \frac{\partial}{\partial x} \left(K_x \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left(K_y \frac{\partial u}{\partial y} \right) = 0 \quad \varepsilon \Omega,$$

and that the corresponding natural boundary condition is

$$n_x K_x \frac{\partial u}{\partial x} + n_y K_y \frac{\partial u}{\partial y} + q + au = 0 \quad \varepsilon \Gamma$$

or simply

$$K_n \frac{\partial u}{\partial n} + q + au = 0 \quad \varepsilon \Gamma,$$

where K_n is the coefficient in the direction of the unit normal, \mathbf{n} . If $K_x = K_y = K$, a constant, show this reduces to the Poisson equation, and Laplaces equation if $Q = 0$.

3. Consider the functional

$$I(u) = \int_0^L \left[EI \left(\frac{d^2u}{dx^2} \right)^2 - 2uQ \right] dx.$$

Show that it yields the ordinary differential equation

$$\frac{d^2}{dx^2} \left(EI \frac{d^2u}{dx^2} \right) = Q,$$

and the natural boundary condition is that

$$- \frac{d}{dx} \left(EI \frac{d^2u}{dx^2} \right) = 0.$$

This is the fourth order equation for the deflection, u , of a beam subject to a transverse load per unit length of Q . Here EI denotes the bending stiffness of the member. Explain why this equation requires four boundary conditions. Note that at any boundary

condition point, one may prescribe u and/or du/dx . If u is not specified, the natural condition is that $d^2u/dx^2 = 0$. They correspond to conditions on the shear and moment, respectively.

4. A one-dimensional Poisson equation is

$$\frac{\partial^2 u}{\partial x^2} = -Q = \frac{105}{2}x^2 - \frac{15}{2}, \quad -1 < x < 1.$$

The exact solution is $u = (35x^4 - 30x^2 + 3) / 8$ when the boundary conditions are

$$u(-1) = 1, \quad \partial u / \partial x(1) = 10.$$

Obtain a finite element solution and sketch it along with the exact solution.

5. A one-dimensional Poisson equation is

$$\frac{\partial^2 u}{\partial x^2} = Q = -6x - \frac{2}{\alpha^2} \left[1 - 2 \left(\frac{x - \beta}{\alpha} \right)^2 \right] \exp \left[- \left(\frac{x - \beta}{\alpha} \right)^2 \right], \quad 0 \leq x \leq 1$$

where β is the interior center position of a local high gradient region, $0 < \beta < 1$, and α is a parameter that governs the amplitude of u in the region, say $\alpha = 0.05$ and $\beta = 0.5$. Obtain a finite element solution and sketch it along with the exact solution for boundary conditions

$$u(0) = \exp \left(- \frac{\beta^2}{\alpha^2} \right), \quad \frac{\partial u}{\partial x}(1) = -3 - 2 \left(\frac{1 - \beta}{\alpha^2} \right) \exp \left[- \left(\frac{1 - \beta}{\alpha} \right)^2 \right],$$

so the exact solution is given by $u = -x^3 + \exp \left[- \left(\frac{x - \beta}{\alpha} \right)^2 \right]$. Note that the source term, Q , may require more integration points than does the evaluation of the square matrix.

6. Buchanan shows that an elastic cable with constant tension, T , resting on an elastic foundation of stiffness k and subjected to a vertical load per unit length of f has a vertical displacement given by the differential equation

$$T \frac{d^2 v}{dx^2} - k v(x) = -f.$$

a) Develop the element matrices for a linear line element, assuming constant T , k and f .
 b) Obtain a finite element solution where $T = 600 \text{ lb}$, $k = 0.5 \text{ lb/in}^2$, $f = 2 \text{ lb/in}$ and where $L = 120 \text{ in}$ is the length of the string where $v = 0$ at each end. [answer: $v_{\max} = 2.54 \text{ in}$]
 c) Replace the model in b) with a half symmetry model where $dv/dx = 0$ at the center ($x = L/2$).

7. Consider a variable coefficient problem given by

$$-[(1+x)u'(x)]' = -1, \quad 0 \leq x \leq 1$$

with the boundary conditions $u(0) = 0 = u(1)$. Obtain a finite element solution and compare it to the exact result $u(x) = x - \ln(1+x)/\ln(2)$.

8. For the differential equation in Problem 2.17 if we have one essential boundary condition of $y(0) = 1$ and one Robin or mixed boundary condition of $y'(1) + y(1) = 0$

the exact solution is $y(x) = (x^4 - 3x^2 - x + 6)/6$ (which is *exact_case 17* in the source library). Obtain a Galerkin finite element solution and compare it to the exact result. Note that this requires mixed element matrices for the point "element" at $x = 1$. (Hint: think about what the 'normal' direction is for each end of a one-dimensional domain.)

9. For the differential equation in Problem 2.17 if we have two Robin or mixed boundary conditions of $y'(0) + y(0) = 0$ and $y'(1) - y(1) = 3$ the exact solution is $y(x) = x^4/6 + 3x^2/2 + x - 1$ (which is *exact_case 18* in the source library). Obtain a Galerkin finite element solution and compare it to the exact result. (Hint: think about what the 'normal' direction is for each end of a one-dimensional domain.)

10. Resolve the heat transfer problem in Fig. 7.7.1 with the external reference temperature increased from 0 to 20 F ($\theta_\infty = 20$). Employ 3 equal length L2 linear elements, and 5 (not 4) nodes with nodes 2 and 3 both at $x = 4/3$. Connect element 1 to nodes 1 and 2 and (incorrectly) connect element 2 to nodes 3 and 4. (a) Solve the incorrect model, sketch the FEA and true solutions, and discuss why the left and right ($x \leq 4/3$ and $x \geq 4/3$) domain FEA solutions behave as they do. (Hint: Think about essential and natural boundary conditions.) (b) Enforce the correct solution by imposing a "multiple point constraint" (MPC) that requires $1 \times t_2 - 1 \times t_3 = 0$. That is, the MPC requires the solution to be continuous at nodes 2 and 3.

11. Implement the exact integral matrices for the linear line element version of Eq. 7.34 and the matrix for recovering the convection heat loss, Eq. 7.38, in the post processing. Apply a small model to the problem in Fig. 7.7.1, but replace the Dirichlet boundary condition at $x = 0$ with the corresponding exact flux $q(0) = 12.86$. Compare the temperature solution with that in Fig. 7.7.1. Why do we still get a solution when we no longer have a Dirichlet boundary condition? (Review Problem 7.1.)

[7–11]

7.11 Bibliography

- [1] Bathe, K.J., *Finite Element Procedures*, Englewood Cliffs: Prentice-Hall (1996).
- [2] Buchanan, G.R., *Finite Element Analysis*, New York: McGraw-Hill (1995).
- [3] Irons, B.M. and Razzaque, A., "Engineering Applications of Numerical Integration in the Stiffness Method," *AIAA Journal*, **4**, pp. 2035–2057 (1966).
- [4] Irons, B.M. and Razzaque, A., "Experience with the Patch Test for Convergence of the Finite Element Method," pp. 557–587 in *Mathematical Foundation of the Finite Element Method*, ed. A.R. Aziz, New York: Academic Press (1972).
- [5] Irons, B.M. and Ahmad, S., *Techniques of Finite Elements*, New York: John Wiley (1980).
- [6] Myers, G.E., *Analytical Methods in Conduction Heat Transfer*, New York: McGraw-Hill (1971).

- [7] Reddy, J.N. and Gartling, D.K., *The Finite Element Method in Heat Transfer and Fluid Dynamics*, Boca Raton: CRC Press (2001).
- [8] Robinson, J., "A Single Element Test," *Comp. Meth. Appl. Mech. Engng.*, **7**, pp. 191–200 (1976).
- [9] Shames, I.H. and Dym, C.L., *Energy and Finite Element Methods in Structural Mechanics*, Pittsburg: Taylor and Francis (1995).
- [10] Wait, R. and Mitchell, A.R., *Finite Element Analysis and Applications*, New York: John Wiley (1985).
- [11] Weaver, W.F., Jr. and Johnston, P.R., *Finite Elements for Structural Analysis*, Englewood Cliffs: Prentice-Hall (1984).
- [12] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method*, 5th Edition, London: Butterworth-Heinemann (2000).