Chapter 2

MATHEMATICAL PRELIMINARIES

2.1 Introduction

The early forms of finite element analysis were based on physical intuition with little recourse to higher mathematics. As the range of applications expanded, for example to the theory of plates and shells, some physical approaches failed and some succeeded. The use of higher mathematics such as variational calculus explained why the successful methods worked. At the same time the mathematicians were attracted by this new field of study. In the last few years the mathematical theory of finite element analysis has grown quite large. Since the state of the art now depends heavily on error estimators and error indicators it is necessary for an engineer to be aware of some basic mathematical topics of finite element analysis. We will consider load vectors and solution vectors, and residuals of various weak forms. All of these require us to define some method to measure these entities. For the above linear vectors (or vector spaces) with discrete coefficients, $\mathbf{V}^T = [\, V_1 \; V_2 \cdots V_n \,]$, we might want to use a measure like the root mean square, *RMS*:

$$RMS^2 \;=\; \frac{1}{n} \sum_{i=1}^{n} V_i^2$$

which we will come to call a norm of the linear vector space. Other quantities vary with spatial position and appear in integrals over the solution domain and/or its boundaries. Thus, we will have to introduce various other norms to "measure" these integral quantities.

The finite element method always involves integrals so it is useful to recall here some integral identities such as Gauss' Theorem (Divergence Theorem):

$$\int_{\Omega} \nabla \cdot \mathbf{u} \, d\Omega \;=\; \int_{\Gamma} \mathbf{u} \cdot \mathbf{n} \, d\Gamma \;=\; \int_{\Gamma} \frac{\partial u}{\partial n} \, d\Gamma$$

which is expressed in Cartesian tensor form as

$$\int_{\Omega} u_{i,i} \; d\,\Omega \;=\; \int_{\Gamma} u_i \; n_i \; d\,\Gamma$$

where there is an implied summation over subscripts that occurs an even number of times

and a comma denotes partial differentiation with respect to the directions that follow it. That is, $(\,)_{,i} = \partial(\,)/\partial x_i$. The above theorem can be generalized to a tensor with any number of subscripts:

$$\int_\Omega A_{ijk\ldots q,r}\, d\Omega = \int_\Gamma A_{ijk\ldots q}\, n_r\, d\Gamma .$$

We will often have need for one of the Green's Theorems:

$$\int_\Omega (\nabla A \cdot \nabla B + A\nabla^2 B)\, d\Omega = \int_\Gamma A \frac{\partial B}{\partial n}\, d\Gamma$$

and

$$\int_\Gamma (A\nabla^2 B - B\nabla^2 A)\, d\Omega = \int_\Gamma (A\nabla B - B\nabla A)\cdot \mathbf{n}\, d\Gamma$$

which in Cartesian tensor form are

$$\int_\Omega (A_{,i}\, B_{,i} + AB_{,ii})\, d\Omega = \int_\Gamma AB_{,i}\, n_i\, d\Gamma$$

and

$$\int_\Omega (AB_{,ii} - BA_{,ii})\, d\Omega = \int_\Gamma (AB_{,i} - BA_{,i})\, n_i\, d\Gamma .$$

We need these relations to derive the Galerkin weak form statements and to manipulate the associated error estimators. Usually, we are interested in removing the highest derivative term in an integral and use the second from last equation in the form

$$\int_\Omega AB_{,ii}\, d\Omega = \int_\Gamma AB_{,i}\, n_i\, d\Gamma - \int_\Omega A_{,i}\, B_{,i}\, d\Omega . \tag{2.1}$$

In one-dimensional applications this process is called integration by parts:

$$\int_a^b pdq = pq\,\Big|_a^b - \int_a^b q\, dp .$$

Error estimators are often based on proofs utilizing inequalities like the Schwarz inequality

$$|\mathbf{a}\cdot\mathbf{b}| \le |\mathbf{a}|\,|\mathbf{b}| \tag{2.2}$$

and the triangle inequality

$$|\mathbf{a} + \mathbf{b}| \le |\mathbf{a}| + |\mathbf{b}| . \tag{2.3}$$

Similar inequalities exist for summations and integrals. Finite element error estimates often use the Minkowski inequality

$$\left[\sum_{i=1}^n |x_i \pm y_i|^p\right]^{1/p} \le \left[\sum_{i=1}^n |x_i|^p\right]^{1/p} + \left[\sum_{i=1}^n |y_i|^p\right]^{1/p}, \quad 1 < p < \infty , \tag{2.4}$$

and the corresponding integral inequality

$$\left[\int_\Omega |x \pm y|^p \, d\Omega\right]^{1/p} \le \left[\int_\Omega |x|^p \, d\Omega\right]^{1/p} + \left[\int_\Omega |y|^p \, d\Omega\right]^{1/p}, \quad 1 < p < \infty. \tag{2.5}$$

We begin the preliminary concepts by introducing linear spaces. These are a collection of objects for which the operations of addition and scalar multiplication are defined in a simple and logical fashion.

2.2 Linear Spaces and Norms

The increased practical importance of error estimates and adaptive methods makes the use of *functional analysis* a necessary tool in finite element analysis. Today's student should consider taking a course in functional analysis, or studying texts such as those of Liusternik [10], Nowinski [12], or Oden [14]. This chapter will only cover certain basic topics. Other related advanced works, such as that of Hughes [9], should also be consulted. We are usually seeking to approximate a more complicated solution by a finite element solution. To develop a feel for the "closeness" or "distance between" these solutions, we need to have some basic mathematical tools. Since the approximation and the true solution vary throughout the spatial domain of interest, we are not interested in examining their difference at an arbitrarily selected point. Instead, we will want to examine integrals of the solutions, or integrals of differences between the solutions. This leads us naturally into the concepts of linear spaces and norms. We will also be interested in integrals of the derivatives of the solution. That will lead us to the Sobolev norm which includes both the function and its derivatives. Consider a set of functions $\phi_1(x)$, $\phi_2(x)$, $\cdots$ $\phi_n(x)$. If the functions can be linearly combined they are called elements of a *linear space*. The following properties hold:

$$\begin{aligned} &\alpha, \ \beta \in R \\ &\phi_1 + \phi_2 = \phi_2 + \phi_1 \\ &(\alpha + \beta)\,\phi = \alpha\,\phi + \beta\,\phi \\ &\alpha\,(\phi_1 + \phi_2) = \alpha\,\phi_1 + \alpha\,\phi_2 . \end{aligned} \tag{2.6}$$

An *inner product*, $< \bullet , \bullet >$, on a real linear space A is a map that assigns to an ordered pair $x, y \in A$ a real number R denoted by $< x, y >$. This process is often represented by the symbolic notation: $< \bullet, \bullet > : A \times A \to R$. It has the following properties

i. $< x, y > = < y, x >$ symmetry

ii. $< \alpha x, y > = \alpha < y, x >$
iii. $< (x + y), z > = < x, z > + < y, z >$
} linearity

iv. $< x, x > \ge 0$ and
$< x, x > = 0$ iff $x = 0$
} positive-definiteness,

The pair $x, y \in A$ are said to be orthogonal if $< x, y > = 0$. A useful property is the Schwarz inequality: $< x, y >^2 \le < x, x > < y, y >$. An inner product also represents an operation such as

$$< u, v > \; = \; \int_{x_1}^{x_2} u(x)\, v(x)\, dx \,. \tag{2.7}$$

Note that when the inner product operations is an integration the symbol $< u, v >$ is often replace by the symbol (u, v) and may be called the *bi-linear form*.

A *norm*, $\| \bullet \|$, on a linear space A is a map of the function to a real number, $\| \bullet \| : A \rightarrow R$, with the properties (for $x, y \in A$ and $\alpha \in R$)

$$\left.\begin{array}{ll} \text{i.} & \| x \| \; \geq \; 0 \quad \text{and} \\ & \| x \| \; = \; 0 \quad \text{iff } x \; = \; 0 \end{array}\right\} \text{ positive-definiteness}$$

$$\begin{array}{ll} \text{ii.} & \| \alpha \; x \| \; = \; | \alpha \; | \; \| x \| \\ \text{iii.} & \| x \; + \; y \| \; \leq \; \| x \| \; + \; \| y \| , \quad \text{triangle inequality} \,. \end{array} \tag{2.8}$$

A *semi-norm*, $| x |$, is defined in a similar manner except that it is positive semi-definite. That is, condition *i* is weakened so we can have $|x| = 0$ for x not zero. A *measure* or *natural norm* of a function x can be taken as the square root of the inner product with itself. This is denoted as

$$\| x \| \; = \; < x, x > ^{\frac{1}{2}} \tag{2.9}$$

2.3 Sobolev Norms *

The $L_2 (\Omega)$ inner product norm involves only the inner product of the functions, and no derivatives: $(u, v) \; = \; \int_{\Omega} uv \; d \, \Omega$ where $\Omega \subset R^n$, $n \geq 1$. Then the norm is

$$\| u \|_{L_2} \; = \; \| u \|_0 \; = \; (u, u)^{\frac{1}{2}} \; = \; [\int_{\Omega} u^2 \; d \; \Omega \;]^{\frac{1}{2}} \,. \tag{2.10}$$

The $H^1 (\Omega)$ inner product and norm includes both the functions and their first derivatives

$$(u, v)_1 \; = \; \int_{\Omega} [\; uv \; + \; \sum_{k=1}^{n} u_{,k} \; v_{,k} \;] \; d \; \Omega$$

where $(\quad)_{,k} \; = \; \partial (\quad) / \partial \; x_k$ and

$$\| u \|_H^1 \; = \; \| u \|_1 \; = \; (u, u)_1^{\frac{1}{2}} \; = \; \left[\int_{\Omega} \left(u^2 \; + \sum_{k=1}^{n} u_{,k}^2 \; \right) d \; \Omega \right]^{\frac{1}{2}} . \tag{2.11}$$

Note $H^0 = L_2$. Likewise, we can extend $H^s (\Omega)$ to include the *S-th* order derivatives and create $\| u \|_s$ norms.

2.4 Dual Problem, Self-Adjointness

One often hears references to a boundary condition as either being an essential or a natural condition. Usually an essential boundary condition simply specifies a value of the primary unknown at a point. However, there is an established mathematical definition of these terms. Consider the differential operator represented by the homogeneous

equations

$$L(u) \;=\; 0 \quad \in \Omega\,. \tag{2.12}$$

We form the inner product of $L(u)$ with another function, say v, to get

$$< L(u), v > \;=\; \int_0^1 u\,\frac{d^2\,v}{dx^2}\,dx\,. \tag{2.13}$$

If we integrate by parts (sometimes repeatedly) we obtain the alternate form

$$< L(u),\, v > \;=\; < u,\, L^*\,(v) > \;+\; \int_\Gamma [\; F(v)\,G(u) \;-\; F(u)\,G^*\,(v)\;]\,d\,\Gamma\,, \tag{2.14}$$

where F and G are differential operators whose forms follow naturally from integration by parts. The operator L^* is the *adjoint* of L. If $L^* = L$ then L is *self-adjoint* and $G^* = G$, also. The $F(u)$ are called the *essential boundary conditions* and $G(u)$ are the *natural boundary conditions*. When $L^* = L$, $F(u)$ is prescribed on Γ_1, and $G(u)$ is prescribed on Γ_2 where $\Gamma = \Gamma_1 \cup \Gamma_2$, $\Gamma_1 \cap \Gamma_2 = \varnothing$. We say $< L(u), u > > 0$ is positive definite *iff* $L^* = L$, and $u \neq 0$. A self-adjoint problem will lead to a set of symmetric bilinear forms and a corresponding set of symmetric algebraic equations for the unknown coefficients in the problem. The weak form given by Eq. 2.14 is also referred to as the *dualproblem*. If both the original weak form and the dual problem are solved it is possible to compute both an upper bound and a lower bound of the error in the approximation. Having both bounds is not always worth the extra computational cost.

To illustrate how to classify the boundary conditions, or to establish a dual problem, consider the model differential equation

$$L(u) \;=\; \frac{d^2u}{dx^2} \qquad x \in\,]0, 1[$$

has the inner product

$$< v,\, L(u) \;> \;=\; \int_0^1 v\;L(u)\;dx \;=\; \int_0^1 v\;\frac{d^2\,u}{dx^2}\;dx\,.$$

Recall integration by parts: $\int_a^b pdq \;=\; pq\,\Big|_a^b \;-\; \int_a^b q\;dp$. Here we let $p = v$ so that $dp \;=\; (dv/dx)\;dx$, and $dq \;=\; (d^2u/dx^2)\;dx$, so $q \;=\; du/dx$, such that

$$< v,\, L(u) \;> \;=\; v\;\frac{du}{dx}\,\Big|_0^1 \;-\; \int_0^1 \frac{du}{dx}\;\frac{dv}{dx}\;dx\,.$$

Integrate by parts again

$$< v,\, L(u) > \;=\; v\,\frac{du}{dx}\,\Big|_0^1 \;-\; [\; u\,\frac{dv}{dx}\,\Big|_0^1 \;-\; \int_0^1 u\,\frac{d^2\,v}{dx^2}\,dx\;]$$

$$=\; < L^*(v),\, u > \;+\; [\; v\,\frac{du}{dx} \;-\; u\,\frac{dv}{dx}\;]\;\Big|_0^1 .$$

Comparing this result to the definitions in Eq. 2.14 we see that the adjoint operator is $L^* = L = d^2()/dx^2$, the essential boundary condition involves $F(v) = 1 * v$, and the natural boundary condition assigns $G() = G^*() = d()/dx$. The original ordinary differential equation requires two boundary conditions. Our usual options are: **a)** give u at $x = 0$ and $x = 1$ and recover du/dx at $x = 0$ and $x = 1$ from the solution, **b)** give u at $x = 0$ and du/dx at $x = 1$ (or vice versa). We compute u for all x and recover du/dx at $x = 0$, **c)** give du/dx at $x = 0$ and $x = 1$. This determines u to within a constant.

2.5 Weighted Residuals

Here we will introduce the concept of approximating the solution to a differential equations by the *method of weighted residuals* (MWR) as it was originally used; on a global basis. That approach requires that we guess the solution over the entire domain and that our guess exactly satisfy the boundary conditions. Then we will introduce the simple but important change that the finite element approach adds to the MWR process. Guessing a solution that satisfies the boundary conditions is very difficult in two- and three-dimensional space, but it is relatively easy in one-dimension. To illustrate a global (or single element solution) consider the following model equation:

$$L(u) \;=\; \frac{d^2\,u}{dx^2} + u + x \;=\; 0\,, \qquad x \in\,]0,1[\tag{2.15}$$

with the essential boundary conditions $u = 0$ at $x = 0$ and $u = 0$ at $x = 1$ so that the exact solution is $u = Sin\,x/Sin\,1 - x$. We want to find a global approximate solution involving constants Δ_i , $1 \le i \le n$ that will lead to a set of n simultaneous equations. For homogeneous essential boundary conditions we usually pick a global product approximation of the form

$$u^* \;=\; g(x)\; f\,(x, \Delta_i\,) \tag{2.16}$$

where $g(x) \equiv 0$ on Γ. Here the boundary is $x = 0$ and $x - 1 = 0$ so we select a form such as $g_1\,(x) \;=\; x\,(1 - x\,)$, or $g_2\,(x) = x - Sin\,x\;/\;Sin\,1$. We could pick $f\,(x, \Delta_i\,)$ as a polynomial $f(x) = \Delta_1 + \Delta_2\,x + \cdots \Delta_n\,x^{(n-1)}$. For simplicity, select $n = 2$ and use $g_1\,(x)$ so the approximate solution is

$$u^*\,(x) \;=\; x\,(\,1 \;-\; x\,)\,(\,\Delta_1 + \Delta_2\;x\,) \;=\; \mathbf{h}\,\mathbf{\Delta}\,. \tag{2.17}$$

Here we will employ the MWR to find the $\mathbf{\Delta}$'s. From Eqs. (2.15 and 17) we see that the residual error at any point is $R(x) \;=\; u\,'' + u + x$, or in expanded form:

$$R(x) \;=\; x + (-2 + x - x^2\,)\,\Delta_1 \;+\; (\,2 - 6x + x^2 - x^3\,)\;\Delta_2 \;\neq\; 0\,. \tag{2.18}$$

For an approximate solution with n constants we can split the residual R into parts including and independent of the Δ_j , say

$$R(x) \;=\; R(x)_0 + \sum_{j=1}^{n} b_j(x)\,\Delta_j \;=\; R_0 + \mathbf{b}\,\boldsymbol{\Delta} \tag{2.19}$$

where **b** is a row matrix and **Δ** is a column vector. Usually R_0 is associated with the source term in the differential equation. Note for future reference that the partial derivatives of the residual with respect to the unknown degrees of freedom are:

$$\partial R/\partial\Delta_1 \;=\; (-2 + x - x^2)\,,\;\; \partial R/\partial\Delta_2 \;=\; (2 - 6x + x^2 - x^3)\,,$$

or in general $\partial R/\partial\Delta_j = b_j(x)$. The residual error will vanish everywhere only if we guess the exact solution. Since that is usually not possible the method of weighted residuals requires that a weighted integral of the residual vanish instead;

$$\int_0^1 R(x)\; w(x)\; dx \;\;\equiv\;\; 0 \tag{2.20}$$

where $w(x)$ is a weighting function. We use n weights to get the necessary system of algebraic equations to find the unknown Δ_j. Substituting Eq. (2.19) gives

$$\int_\Omega R\, w_k\; d\,\Omega \;\;=\;\; \int_\Omega \left(R_0 + \sum_{j=1}^{n} b_j(x)\,\Delta_j \right) w_k\; d\,\Omega \;=\; 0_k\,, \quad 1 \;\le\; k \;\le\; n$$

or

$$\sum_{j=1}^{n} \int_\Omega b_j(x)\; w_k(x)\; \Delta_j\; d\,\Omega \;=\; -\int_\Omega R_0(x)\; w_k(x)\; d\,\Omega\,, \quad 1 \;\le\; k \;\le\; n\,. \tag{2.21}$$

In matrix form this system of equations is written as:

$$\begin{array}{ccc} \mathbf{S} & \Delta & = \;\mathbf{C} \\ n \;\times\; n & \mathrm{n} \;\times\; 1 & \mathrm{n} \;\times\; 1\,. \end{array} \tag{2.22}$$

Usually we call **S** and **C** the stiffness matrix and source vector, respectively. Clearly, there are many ways to pick the weighting functions, w_k. Mathematical analysis and engineering experience have lead to the following five most common choices of the weights used in various weighted residual methods:

A) Collocation Method: For this method we force the residual error to vanish at n arbitrarily selected points. Thus, we select

$$w(x) \;=\; \delta(x - x_k)\,, \qquad 1 \;\le\; k \;\le\; n \tag{2.23}$$

where the Dirac Delta distribution $\delta(x - x_k)$ which has the properties

$$\delta(x - x_k) \;\;=\;\; \begin{cases} 0 & x \neq x_k \\ \infty & x \;=\; x_k \end{cases}$$

and

$$\int_{-\infty}^{\infty} \delta(x - x_k)\; dx \;\;=\;\; \int_{x_k - a}^{x_k + a} \delta(x - x_k)\; dx \;=\; 1\,.$$

and for any function $f(x)$ continuous at x_k

$$\int_{-\infty}^{\infty} \delta(x - x_k)\, f(x)\, dx \;=\; \int_{x_k - a}^{x_k + a} \delta(x - x_k)\, f(x)\, dx \;=\; f(x_k). \qquad (2.24)$$

By inspection this reduces Eq. (2.21) to simply

$$\sum_{j=1}^{n} b_j(x_k)\, \Delta_j \; - R_0(x_k), \;\; 1 \le k \le n.$$

Our problem is that we have an infinite number of choices for the collocation points, x_k. For $n = 2$, we could pick two points where R is large, or the third point, or the Gauss points, etc. Pick the two collocation points as $x_1 = 1/4$ and $x_2 = 1/2$; then

$$\begin{bmatrix} \frac{29}{16} & -\frac{35}{64} \\ \frac{7}{4} & \frac{7}{8} \end{bmatrix} \begin{Bmatrix} \Delta_1 \\ \Delta_2 \end{Bmatrix} \;=\; \begin{Bmatrix} \frac{1}{4} \\ \frac{1}{2} \end{Bmatrix}$$

is our unsymmetric algebraic system. Since the essential boundary conditions have already been satisfied by the assumed solution we can solve these equations without additional modifications. Here we obtain $\Delta_1 = 6/31$ and $\Delta_2 = 40/217$ so that our first approximate solution is $u^* = x(1 - x)(42 + 40x)/217$. Selected interior results compared to the exact solution are:

x	u	u^*
1/4	0.044	0.045
1/2	0.070	0.071
3/4	0.060	0.062

Note that $u(x_k) - u^*(x_k) \neq 0$ even though $R(x_k) = 0$. That is, the error in the differential equation is zero at these collocation points, but the error in the solution is not zero. This can be viewed as similar to a finite difference solution.

B) Least Squares Method: For the n equations pick

$$\int_0^1 R(x)\, w_i(x)\, dx \;=\; 0, \qquad 1 \le i \le n$$

with the weights defined as

$$w_i(x) \;=\; \frac{\partial R(x)}{\partial \Delta_i} \;=\; b_i(x), \qquad (2.25)$$

from Eq.(2.19). This choice is equivalent to solving the minimization problem:

$$\int_0^1 R^2(x)\, dx \quad \rightarrow \quad \text{stationary (minimum)}. \qquad (2.26)$$

Equation (2.26) means in this case Eq. (2.21) becomes

$$\sum_{j=1}^{n} \int_{\Omega} b_j(x)\, b_i(x)\, \Delta_j \, d\Omega \;=\; -\int_{\Omega} R_0(x)\, b_i(x)\, d\Omega, \quad 1 \le i \le n.$$

For this example

$$\int_0^1 R(x)\,\frac{\partial R}{\partial \Delta_1}\,dx \;=\; 0\,, \qquad \int_0^1 R(x)\,\frac{\partial R}{\partial \Delta_2}\,dx \;=\; 0$$

and substitutions from Eq. (2.18) gives

$$\frac{202}{60}\,\Delta_1 \;+\; \frac{101}{60}\,\Delta_2 \;=\; \frac{55}{60}$$

$$\frac{101}{60}\,\Delta_1 \;+\; \frac{393}{105}\,\Delta_2 \;=\; \frac{57}{60}\,.$$

It should be noted from Eqs. (2.19, 21, 25) that this procedure yields a square matrix which is always symmetric. Solving gives $\Delta_1 = 0.188$, $\Delta_2 = 0.170$ and selected results at the three interior points of : 0.043, 0.068, and 0.059, respectively.

C) Galerkin Method: The concept here is to make the residual error orthogonal to the functions associated with the spatial influence of the constants. That is, let

$$u^*(x) \;=\; g(x)\; f\,(x, \Delta_i\,) \;=\; \sum_{i=1}^{n} h_i\,(x)\,\Delta_i\,.$$

Here the h_i term defines how we have assumed the contribution from Δ_i will vary over space. Here for $n = 2$ and $h_1 = (x - x^2)$ and $h_2 = (x^2 - x^3)$, we set

$$w_i\,(x) \;\equiv\; h_i\,(x) \tag{2.27}$$

so Eq. (2.21) simplifies to

$$\sum_{j=1}^{n} \int_\Omega b_j\,(x)\,h_i\,(x)\,\Delta_j\;d\Omega \;=\; -\int_\Omega R_0\,(x)\,h_i\,(x)\;d\Omega\,, \qquad 1 \;\le\; i \;\le\; n\,. \tag{2.28}$$

and for this specific example we require

$$\int_0^1 R(x)\,h_1\,(x)\;dx \;=\; 0\,, \qquad \int_0^1 R(x)\,h_2\,(x)\;dx \;=\; 0$$

and Eq. (2.18) yields

$$\frac{3}{10}\,\Delta_1 \;+\; \frac{3}{20}\,\Delta_2 \;=\; \frac{1}{12}$$

$$\frac{3}{20}\,\Delta_1 \;+\; \frac{13}{105}\,\Delta_2 \;=\; \frac{1}{20}$$

which is again symmetric (for the self-adjoint equation). Solving gives degree of freedom values of $\Delta_1 = 71/369$, $\Delta_2 = 7/41$ and selected results at the three interior points of : 0.044, 0.070, and 0.060, respectively.

D) Method of Moments : Pick a spatial coordinate "lever arm" as a weight :

$$w_i\,(x) \;\equiv\; x^{(i-1)} \tag{2.29}$$

so that in the current one-dimensional example

$$\int_0^1 R(x)\, x^0\, dx \;=\; 0\,, \qquad \int_0^1 R(x)\, x^1\, dx \;=\; 0 \tag{2.30}$$

gives the algebraic system

$$\frac{11}{6}\,\Delta_1 \;+\; \frac{11}{12}\,\Delta_2 \;=\; \frac{1}{2}$$

$$\frac{11}{12}\,\Delta_1 \;+\; \frac{19}{20}\,\Delta_2 \;=\; \frac{1}{3}$$

with the solution $\Delta_1 = 122/649$, $\Delta_2 = 110/649$ and selected results at the three interior points of: 0.043, 0.068, and 0.059, respectively. This method usually yields an unsymmetrical system. It is popular in certain physics applications.

E) Subdomain Method: For this final method we split the solution domain, Ω, into n arbitrary non-overlapping subdomains that completely fill the space such that

$$\Omega \;=\; \bigcup_{k=1}^{n} \Omega_k \tag{2.31}$$

Then we define

$$w_k\,(x) \;\equiv\; 1 \qquad \text{for} \quad x \in \Omega_k\,. \tag{2.32}$$

and it is zero elsewhere. This makes the residual error vanish on each of n different regions. Here $n = 2$, so we arbitrarily pick $\Omega_1 =]0, \frac{1}{2}[$ and $\Omega_2 =]\frac{1}{2}, 1[$. Then

$$\int_{\Omega_1} R(x)\, dx \;=\; 0\,, \qquad \int_{\Omega_2} R(x)\, dx \;=\; 0 \tag{2.33}$$

yields the unsymmetric algebraic system

$$\begin{bmatrix} \frac{11}{6} & \frac{11}{6} \\ \frac{11}{22} & \frac{19}{20} \end{bmatrix} \begin{Bmatrix} \Delta_1 \\ \\ \Delta_2 \end{Bmatrix} \;=\; \begin{Bmatrix} \frac{1}{2} \\ \\ \frac{1}{3} \end{Bmatrix}.$$

This results in $\Delta_1 = 122/649$, $\Delta_2 = 110/649$ and selected results at the three interior points of: 0.043, 0.068, and 0.059, respectively.

The above examples illustrate how analytical approximations can be obtained for differential equations. These approximate methods offer some practical advantages. Instead of solving a differential equation we are now presented with the easier problem of solving an integral relation. The weighted residual procedure is valid of any number of spatial dimensions. The procedure is valid for any shaped domain Ω. It allows non-homogeneous coefficients. That is, the coefficient multiplying the derivatives in the differential operator L can vary with location. Note that so far we have not yet made any references to finite element methods. Later, you may look back on these examples as special cases of a single element solution. These simple hand examples could have been solved with the elementary matrix inversion routines like INVERT_2BY2 and INVERT_3BY3 that are included on the source code Web site. In practice, inversions are much too computationally expressive, and one must solve the equations by iterative

methods or by a factorization process such as the process outlined in Fig. 2.5.1. By starting with a triangular matrix the substitution processes have only one unknown per row. The factored triangular arrays are stored in the locations of the original square matrix. The actual algorithm details are given in a later chapter. Practical implementations of direct solvers must account for sparse array storage options.

2.6 Boundary Condition Terms

If the assumed solution does not satisfy the essential boundary conditions then we must extend the assumed approximate solution to include additional constants to be used to satisfy the essential boundary conditions. Usually these conditions are invoked prior to or during the solution of the algebraic equations. Sometimes they are satisfied only in a weak sense. To illustrate the algebraic procedure that is usually used we will solve the ODE in Eq. (2.15) with an approximation that does not initially satisfy the boundary condition at $x = 0$. To apply the boundary condition after we have selected an approximate solution we will use Eq. (2.16) and pick $g(x) = (1 - x)$ so that only the boundary condition at $x = 1$ is satisfied in advance. We can add another constant to $f(x)$ to allow any boundary condition at $x = 0$ to be applied:

$$f(x) = \Delta_1 + \Delta_2 x + \Delta_3 x^2 .$$

The new residual error for any x is

$$R = x + (1 - x)\,\Delta_1 + (x + 2 - x^2)\,\Delta_2 + (x^2 + 2 - 6x - x^3)\,\Delta_3 .$$

Since we now have three unknown degrees of freedom, Δ, we must have three weighted residual equations. For simplicity we will choose the collocation method and pick three equally spaced collocation points. Evaluating the residual at the quarter points and multiplying by the common denominator gives the three equations

$$\begin{bmatrix} -48 & 116 & -35 \\ -32 & 112 & 56 \\ -16 & 116 & 151 \end{bmatrix} \begin{Bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \end{Bmatrix} = \begin{Bmatrix} 16 \\ 32 \\ 48 \end{Bmatrix}.$$

Note that since we know u at $x = 0$ these unknowns are not independent. Substituting $x = 0$ into our approximate solution and equating it to the assign boundary value there gives $u(0) = \Delta_1$. We call this an *essential boundary condition* on Δ_1. There are an infinite number of possible boundary conditions and we gain flexibility by allowing extra constants to satisfy them. Since Δ_1 will be a known number only the last two rows are independent for determining the remaining terms in Δ. Note that the first column of numbers, in the last two rows, is now multiplied by a known value and thus they can be carried to the right hand side to give the reduced algebraic system for the independent Δ:

$$\begin{bmatrix} 112 & 56 \\ 116 & 151 \end{bmatrix} \begin{Bmatrix} \Delta_2 \\ \Delta_3 \end{Bmatrix} = \begin{Bmatrix} 32 \\ 48 \end{Bmatrix} + \begin{Bmatrix} 32 \\ 16 \end{Bmatrix} \Delta_1 .$$

An equivalent matrix modification routine in the MODEL code deletes the redundant coefficients, but keeps the matrix the same size to avoid re-ordering all the coefficients as done above. For the common original boundary condition of $u(0) = 0$, we have $\Delta_1 = 0$

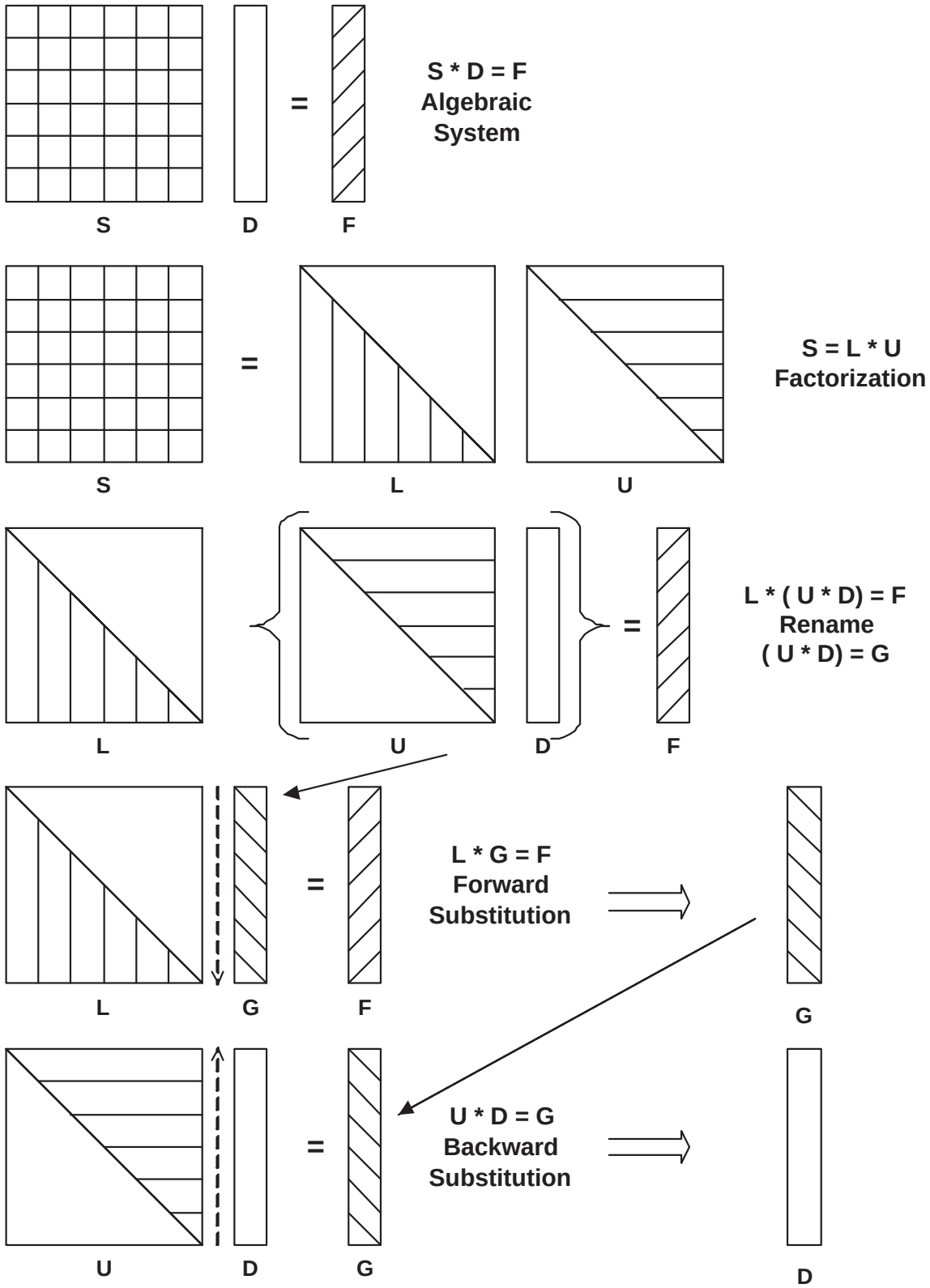

Figure 2.5.1 Steps in the factorization process

and the changes to the right hand side (RHS) are not necessary. But the above form also allows us the option of specifying any non-zero boundary condition we need. Using the zero value gives a solution of $\Delta_2 = 0.2058$ and $\Delta_3 = 0.1598$. The resulting values at the interior quarter points are 0.046, 0.071, and 0.061, respectively. These compare well with the previous results.

If the second boundary condition had been applied other than at $x = 0$ then we would have a more complicated relation between the Δ. For example, assume we move the boundary condition to $x = 0.5$. Then evaluating the approximate solution there yields

$$u(0.5) \;=\; 0.5\Delta_1 + 0.25\Delta_2 \;+\; 0.125\Delta_3$$

which is called a *linear constraint equation* on Δ, or a *multipoint constraint* (MPC). In other words we would have to solve the weighted residual algebraic system subject to a linear constraint. This is a fairly common situation in practical design problems and adaptive analysis procedures. The computational details for enforcing the above essential boundary conditions are discussed in detail later.

2.7 Adding More Unknowns

Since the exact solution of the model problem is not a polynomial our global polynomial approach can never yield an exact solution. However, we can significantly improve the accuracy by adding more unknown coefficients to the expansion in Eq. (2.17). In matrix notation the original global Galerkin matrices become

$$\mathbf{S}^e \;=\; \int_L \mathbf{h}^T\, \mathbf{b}\; dx, \qquad \mathbf{C}^e \;=\; \int_L \mathbf{h}^T\, Q(x)\; dx$$

where $Q(x) = x$ is the source term, $\mathbf{b} = \mathbf{h}'' + \mathbf{h}$ comes from the differential operator acting on u, and where a prime denotes a derivative. Likewise, using Least Squares:

$$\mathbf{S}^e \;=\; \int_L \mathbf{b}^T\, \mathbf{b}\; dx, \qquad \mathbf{C}^e \;=\; \int_L \mathbf{b}^T\, Q(x)\; dx$$

Here we see that the Least Squares square matrix will always be symmetric, but the Galerkin form may not be. As we add more unknown coefficients we just increase the size of the functions in **h**, and thus in **b**, and increase the number of integration points to account for the higher degree polynomials occurring in the matrices. A disadvantage of adding more unknowns to a global solution is that the unknown parameters are fully coupled to each other. That means the algebraic equations to be solved are fully populated, and thus very expensive to solve. The finite element method will lead to very sparse equations that are efficient to solve.

2.8 Numerical Integration

Since numerical integration simply replaces an integral with a special summation this approach has the potential for automating all the above integrals required by the MWR. Then we can include thousands of unknown coefficients, Δ_i , in our test solution. Here we are dealing with polynomials. It is well known that in one-dimension Gaussian quadrature with n terms will exactly integrate a polynomial of order $(2n - 1)$. Gauss proved that this is the minimum number of points that can be used in a summation to

Application Dependent Software

Term / Process	Required	Optional
Generate matrices		
Differential operator $\mathbf{S}^{E}_{K}$	ELEM_SQ_MATRIX my_el_sq_inc	APPLICATION_B_MATRIX my_b_matrix_inc APPLICATION_E_MATRIX my_e_matrix_inc
Volumetric Source $\mathbf{C}^{E}_{Q}$	either	ELEM_COL_MATRIX my_el_col_inc or ELEM_SQ_MATRIX my_el_sq_inc
Mixed or Robin BC $\mathbf{S}^{B}_{h}$, $\mathbf{C}^{B}_{t}$	MIXED_SQ_MATRIX my_mixed_sq_inc	
Boundary Flux $\mathbf{C}^{B}_{F}$	SEG_COL_MATRIX my_seg_col_inc	EXACT_NORMAL_FLUX my_exact_normal_flux_inc
Save for post-processing		ELEM_POST_DATA my_el_post_inc
Post-process element		POST_PROCESS_ELEM my_post_el_inc
Energy norm error estimate	APPLICATION_B_MATRIX my_b_matrix_inc APPLICATION_E_MATRIX my_e_matrix_inc	
Use an exact solution		
Exact essential BC		EXACT_SOLUTION my_exact_inc
List exact solution		EXACT_SOLUTION my_exact_inc
List exact fluxes		EXACT_SOLUTION_FLUX my_exact_flux_inc
Use exact source		EXACT_SOURCE my_exact_source_inc

Figure 2.8.1 User software interfaces in MODEL

yield the exact results. Therefore, it is the most efficient method available for integrating polynomials. Thus, we could replace the above integrals with a two-point Gauss rule. (This will be considered in full detail later in Sec. 5.4.) For example, in the Galerkin approach the source term is

$$\int_0^1 x\, h_1(x)\, dx \;=\; \sum_{j=1}^{n} x_j\, h_1(x_j)\, w_j \tag{2.34}$$

where the x_j and w_j are tabulated. For $n = 2$ on the domain $\Omega =]0,1[$ we have $w_1 = w_2 = 1/2$ and $x_j = (1 \pm 1/\sqrt{3})/2$, or $x_1 = 0.2113325$ and $x_2 = 0.788675$. So

$$\int_0^1 x\,(x - x^2)\, dx \;=\; \sum_{j=1}^{n} (x_j^2 - x_j^3)\, w_j \;=\; \sum_{j=1}^{2} x_j^2\,(1 - x_j)\, w_j$$

$$= [\,(0.2113248)^2\,(0.7886751)\,1/2 \;+\; (0.7886751)^2\,(0.2113248)\,1/2\,]$$

$$= (0.16666667)\,1/2 \;=\; 0.083333\,.$$

If we had an infinite word length machine this process would yield the exact value of 1/12 which was previously found in Eq. (2.28).

A typical partial implementation of these global Galerkin and Least Squares procedures will be illustrated with the MODEL program. Only a very small part of it changes for each application. Every application requires that we formulate a square matrix. That is done in subroutine ELEM_SQ_MATRIX, which also allows the optional calculation of an element column matrix. A number of prior applications are supplied in a library form and will be discussed later. The coding for a totally new application is usually supplied by an "include file" that the compiler inserts into the necessary subprogram. Figure 2.8.1 shows all of the user interfaces that we will use in this book. Note that for educational purposes it includes access to selected exact solutions so they can be compared to the finite element model solution and the error estimator to be consider later. By using the "keyword" controls in a data file MODEL allocates space for the most commonly needed items in a finite element analysis. As we find need for such items we will declare how they interface to subroutine ELEM_SQ_MATRIX, and others. Figure 2.8.2 lists the portion of the interface that will be used here.

The coding of the above problems, by numerical integration, are shown in Figs. 2.8.3 and 2.8.4, respectively. The results agree well, as do all of our weighted residual solutions. The global Galerkin and Least Squares results are listed in Fig. 2.8.5. Plotting the resulting solutions shows very similar curves from all five approaches to the MWR.


```
          Interface from MODEL to ELEM_SQ_MATRIX, 1
Note: MODEL requires strong typing (implicit none).  Any item not
defined here (and later) in the interface must have its variable
type and size defined by the user.
Type   Status Name   Remarks                          (keyword)
INTEGER (IN) DP      Double precision kind for this hardware
INTEGER (IN) LT_GEOM Number of element type geometric nodes
INTEGER (IN) LT_FREE Number of element type unknowns
INTEGER (IN) LT_N    Number of element type solution nodes (el_nodes)
INTEGER (IN) LT_PARM Parametric dimension of element type
INTEGER (IN) LT_QP   Number of element type quadrature points
INTEGER (IN) N_SPACE Physical space dimension of problem (space)
REAL(DP) (IN) COORD (LT_N, N_SPACE)  Element type coordinates
REAL(DP) (IN) PT    (LT_PARM, LT_QP) Quadrature parametric points
REAL(DP) (IN) WT    (LT_QP)          Quadrature parametric weights
REAL(DP) (OUT) C   (LT_FREE)          Element column matrix
REAL(DP) (OUT) DGH (N_SPACE, LT_N)    Global derivatives of H
REAL(DP) (OUT) DLH (LT_PARM, LT_N)    Local  derivatives of H
REAL(DP) (OUT) G   (LT_GEOM)          Geometry interpolation array
REAL(DP) (OUT) H   (LT_N)             Solution interpolation array
REAL(DP) (OUT) S   (LT_FREE, LT_FREE) Element square matrix
GET_G_AT_QP    Form G array at quadrature point
GET_H_AT_QP    Form H array at quadrature point
GET_DLH_AT_QP  Form DLH array at quadrature point
```

Figure 2.8.2 User interface to ELEM_SQ_MATRIX, part 1

2.9 Integration By Parts

The use of integration by parts will be very important in most finite element Galerkin methods. From the matrix definitions in the previous sections we see that the Least Square process involves the same order derivative in both terms in the matrix product in the square matrix and thus can not benefit from integration by parts. However, in the Galerkin square matrix since $\mathbf{b} = \mathbf{h}'' + \mathbf{h}$, the first product involves $\mathbf{h}$ and $\mathbf{h}''$ so integration by parts can be applied to that one matrix product. Returning to the original scalar form causing that term we see:

$$\int_L w\, u''\, dx = w\, u' \Big|_0^L - \int_L w' u'\, dx\,. \tag{2.35}$$

Here the assumed solution is zero at the two ends so the appearance of the boundary terms is not clearly important in this global analysis. But in finite element analysis, where we will have extra unknown coefficients at the end points, they will be very important and yield physically significant reaction recovery data. When we utilize the above integration by parts, and change all the signs, the previous coding in Fig. 2.8.3 changes to that in Fig. 2.9.1. Note that the square matrix is now clearly symmetric, and we no longer need the storage array for the second derivative of h. The numerical results are identical to the original ones given in Fig. 2.8.3 above. The full cubic approximation is seen in Fig. 2.9.2. If we had only one degree of freedom ($\Delta_2 = 0$) this would reduce to a


```
! ...  Partial Global Access Arrays                                  ! 1
 REAL(DP) :: C (LT_FREE), S (LT_FREE, LT_FREE) ! Results      ! 2
 REAL(DP) :: PT (LT_PARM, LT_QP), WT (LT_QP)   ! Quadratures  ! 3
 REAL(DP) :: H (LT_N), DGH (N_SPACE, LT_N)     ! Solution     ! 4
 REAL(DP) :: G (LT_GEOM)                       ! Geometry     ! 5
 REAL(DP) :: COORD (LT_N, N_SPACE)             ! Coordinates  ! 6
                                                              ! 7
! ...  Partial Notations List                                  ! 8
! COORD     = SPATIAL COORDINATES OF ELEMENT'S NODES           ! 9
! DGH       = GLOBAL DERIVATIVES OF INTERPOLATION FUNCTIONS    !10
! G         = GEOMETRIC INTERPOLATION FUNCTIONS                !11
! H         = SCALAR INTERPOLATION FUNCTIONS                   !12
! LT_FREE   = NUMBER OF DEGREES OF FREEDOM                     !13
! LT_GEOM   = NUMBER OF GEOMETRY NODES                         !14
! LT_PARM   = DIMENSION OF PARAMETRIC SPACE                    !15
! LT_QP     = NUMBER OF QUADRATURE POINTS                      !16
! LT_N      = NUMBER OF NODES PER ELEMENT                      !17
! N_SPACE   = DIMENSION OF PHYSICAL SPACE                      !18
! PT        = QUADRATURE COORDINATES                           !19
! WT        = QUADRATURE WEIGHTS                               !20
! ...         see full notation file                           !21
                                                               !22
! ..........................................................   !23
! **  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW **    !24
! ..........................................................   !25
! Define any new local array or variable types                 !26
!                                                              !27
!      GLOBAL (SINGLE ELEMENT) Galerkin MWR FOR ODE            !28
!   U,xx + U + X = 0, U(0)=0=U(1), U = Sin(x)/Sin(1) - x       !29
!             Without integration by parts                     !30
                                                               !31
 REAL(DP) :: D2GH (1, 1:2)    ! Second global derivative       !32
 REAL(DP) :: DL, DX_DR, X_Q   ! Length, Jacobian, Position     !33
 INTEGER  :: IQ               ! Loops                          !34
                                                               !35
 DL    = COORD (LT_N, 1) - COORD (1, 1)  ! LENGTH             !36
 DX_DR = DL / 2.                         ! CONSTANT JACOBIAN  !37
 S = 0.d0; C = 0.d0                      ! ZERO SUMS          !38
                                                               !39
 DO IQ = 1, LT_QP        ! LOOP OVER QUADRATURES              !40
                                                               !41
!        GET GEOMETRIC INTERPOLATION FUNCTIONS, AND X-COORD    !42
  G   = GET_G_AT_QP (IQ)                        ! parametric   !43
  X_Q = DOT_PRODUCT (G, COORD (1:LT_GEOM, 1)) ! x at point     !44
                                                               !45
!    GLOBAL INTERPOLATION, 1st & 2nd GLOBAL DERIVATIVES        !46
  H    (:)   = (/ (X_Q - X_Q**2), (X_Q**2 - X_Q**3)  /)        !47
  DGH  (1,:) = (/ (1 - 2*X_Q),    (2*X_Q - 3*X_Q**2) /)        !48
  D2GH (1,:) = (/ (-2),           (2 - 3*X_Q)        /)        !49
                                                               !50
  C = C - H * X_Q * WT (IQ) * DX_DR  ! SOURCE, from Q(x)       !51
                                                               !52
!          SQUARE MATRIX ( ? SYMMETRIC ? )                     !53
  S = S + ( MATMUL (TRANSPOSE(D2GH), H)          & ! from u"   !54
          + OUTER_PRODUCT (H, H)) * WT (IQ) * DX_DR ! from u   !55
 END DO  ! QUADRATURE                                          !56
!  Outer product C_sub_jk = A_sub_j * B_sub_k                  !57
!  End of application dependent code                           !58
```

Figure 2.8.3 A global Galerkin implementation


```
! ...  Partial Global Access Arrays                                  ! 1
 REAL(DP) :: C (LT_FREE), S (LT_FREE, LT_FREE) ! Results     ! 2
 REAL(DP) :: PT (LT_PARM, LT_QP), WT (LT_QP)   ! Quadratures ! 3
 REAL(DP) :: H (LT_N), DGH (N_SPACE, LT_N)     ! Solution    ! 4
 REAL(DP) :: G (LT_GEOM)                       ! Geometry    ! 5
 REAL(DP) :: COORD (LT_N, N_SPACE)             ! Coordinates ! 6
                                                             ! 7
! ...  Partial Notations List                                 ! 8
! COORD   = SPATIAL COORDINATES OF ELEMENT'S NODES           ! 9
! DGH     = GLOBAL DERIVATIVES OF INTERPOLATION FUNCTIONS    !10
! G       = GEOMETRIC INTERPOLATION FUNCTIONS                !11
! H       = SCALAR INTERPOLATION FUNCTIONS                   !12
! LT_FREE = NUMBER OF DEGREES OF FREEDOM                     !13
! LT_GEOM = NUMBER OF GEOMETRY NODES                         !14
! LT_PARM = DIMENSION OF PARAMETRIC SPACE                    !15
! LT_QP   = NUMBER OF QUADRATURE POINTS                      !16
! LT_N    = NUMBER OF NODES PER ELEMENT                      !17
! N_SPACE = DIMENSION OF PHYSICAL SPACE                      !18
! PT      = QUADRATURE COORDINATES                           !19
! WT      = QUADRATURE WEIGHTS                               !20
! ...       see full notation file                           !21
                                                             !22
! .....................................................      !23
! **  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW **  !24
! .....................................................      !25
! Define new local array or variable types, then statements  !26
!                                                            !27
!    GLOBAL (SINGLE ELEMENT) LEAST SQUARE METHOD FOR ODE     !28
!    U,xx + U + X = 0, U(0)=0=U(1), U = Sin(x)/Sin(1) - x    !29
                                                             !30
 REAL(DP) :: DL, DX_DR, X_IQ ! Length, Jacobian, Position    !31
 REAL(DP) :: D2GH (1, 2)     ! Second derivative             !32
 REAL(DP) :: F (2)           ! H'' + H, Work space           !33
 INTEGER  :: IQ              ! Loops                         !34
                                                             !35
 DL    = COORD (LT_N, 1) - COORD (1, 1) ! LENGTH             !36
 DX_DR = DL / 2.                        ! CONSTANT JACOBIAN  !37
 S = 0.d0; C = 0.d0                     ! ZERO SUMS          !38
                                                             !39
 DO IQ = 1, LT_QP        ! LOOP OVER QUADRATURES             !40
                                                             !41
!       GET GEOMETRIC INTERPOLATION FUNCTIONS, AND X-COORD   !42
   G   = GET_G_AT_QP (IQ)                       ! parametric  !43
   X_Q = DOT_PRODUCT (G, COORD (1:LT_GEOM, 1)) ! x at point  !44
                                                             !45
!       GLOBAL INTERPOLATION, 1st & 2nd GLOBAL DERIVATIVES   !46
   H    (:)   = (/ (X_Q - X_Q**2), (X_Q**2 - X_Q**3)  /)     !47
   DGH  (1,:) = (/ (1 - 2*X_Q),    (2*X_Q - 3*X_Q**2) /)     !48
   D2GH (1,:) = (/ (-2),           (2 - 6*X_Q)        /)     !49
   F    (:)   = D2GH (1,:) + H (:)                           !50
                                                             !51
   C = C - F * X_Q * WT (IQ) * DX_DR ! SOURCE, from Q(x)     !52
                                                             !53
!        SQUARE MATRIX, from U" and U, SYMMETRIC             !54
   S = S + OUTER_PRODUCT (F, F) * WT (IQ) * DX_DR            !55
 END DO  ! QUADRATURE                                        !56
!  Outer product C_sub_jk = A_sub_j * B_sub_k                !57
!  End of application dependent code                         !58
```

Figure 2.8.4 A global least squares implementation


```
U,xx + U + X = 0, U(0)=0=U(1), Exact U = Sin(x)/Sin(1) - x    ! 1
                                                              ! 2
Single cubic element (Global) Galerkin Solution:              ! 3
-----------------------------------------------              ! 4
                                                              ! 5
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                    ! 6
  NODE, 1 COORDINATES,  1 PARAMETERS.                         ! 7
     1  0.00000E+00     1.92412E-01                           ! 8
     2  1.00000E+00     1.70732E-01                           ! 9
                                                             !10
** ELEMENT GAUSS POINT  RESULTS **                           !11
ELEM    X       EXACT         FEA      GRADIENT   FE_GRADIENT !12
1   0.000  0.0000E+00  0.0000E+00   1.88395E-01   1.92412E-01 !13
1   0.069  1.3014E-02  1.3198E-02   1.85532E-01   1.86932E-01 !14
1   0.330  5.5092E-02  5.5001E-02   1.24268E-01   1.22321E-01 !15
1   0.670  6.7974E-02  6.7835E-02  -6.85032E-02  -6.65570E-02 !16
1   0.931  2.2476E-02  2.2697E-02  -2.90078E-01  -2.91477E-01 !17
1   1.000  0.0000E+00  0.0000E+00  -3.57907E-01  -3.63144E-01 !18
                                                             !19
Single cubic element (Global) Least square Solution:         !20
---------------------------------------------------          !21
                                                             !22
***  OUTPUT OF RESULTS IN NODAL ORDER  ***                   !23
  NODE, 1 COORDINATES,  1 PARAMETERS.                        !24
     1  0.00000E+00     1.87542E-01                          !25
     2  1.00000E+00     1.69471E-01                          !26
                                                             !27
** ELEMENT GAUSS POINT  RESULTS **                          !28
ELEM    X       EXACT         FEA      GRADIENT   FE_GRADIENT !29
1   0.000  0.0000E+00  0.0000E+00   1.88395E-01   1.87542E-01 !30
1   0.034  6.3536E-03  6.3053E-03   1.87718E-01   1.85742E-01 !31
1   0.169  3.0952E-02  3.0426E-02   1.71385E-01   1.66831E-01 !32
1   0.381  6.0872E-02  5.9426E-02   1.03316E-01   1.00101E-01 !33
1   0.619  7.0522E-02  6.8960E-02  -3.23143E-02  -2.98400E-02 !34
1   0.831  4.6834E-02  4.6193E-02  -1.98511E-01  -1.93234E-01 !35
1   0.966  1.1519E-02  1.1461E-02  -3.24515E-01  -3.22039E-01 !36
1   1.000  0.0000E+00  0.0000E+00  -3.57907E-01  -3.57013E-01 !37
```

Figure 2.8.5 Global MWR solutions and gradients.

quadratic approximation with much higher error as seen in Fig. 2.9.3. Increasing the number of degrees of freedom quickly decreases the error to the point that it can not be seen, but can be computed by an error estimator.

2.10 Finite Element Model Problem

In order to extend the previous introductory concepts on the global MWR to the more powerful finite element method consider the same one-dimensional model problem as our first example. Recall that the differential equation of interest, Eq. (2.15), is

$$L(u) \;=\; \frac{d^2\,u}{dx^2} + u + Q(x) \;=\; 0\,, \qquad x \in \,]0, L[$$

on the closed domain, $x \in \,]\,0, L\,[$, and is subjected to two boundary conditions to yield a unique solution. Here $Q(x) = x$ denotes a source term per unit length, as before. The corresponding governing integral statement to be used for the finite element model is


```
! ...  Partial Global Access Arrays                                  ! 1
 REAL(DP) :: C (LT_FREE), S (LT_FREE, LT_FREE) ! Results            ! 2
 REAL(DP) :: PT (LT_PARM, LT_QP), WT (LT_QP)   ! Quadratures        ! 3
 REAL(DP) :: H (LT_N), DGH (N_SPACE, LT_N)     ! Solution           ! 4
 REAL(DP) :: G (LT_GEOM)                       ! Geometry           ! 5
 REAL(DP) :: COORD (LT_N, N_SPACE)             ! Coordinates        ! 6
                                                                    ! 7
! ...  Partial Notations List                                       ! 8
! COORD   = SPATIAL COORDINATES OF ELEMENT'S NODES                  ! 9
! DGH     = GLOBAL DERIVATIVES OF INTERPOLATION FUNCTIONS           !10
! G       = GEOMETRIC INTERPOLATION FUNCTIONS                       !11
! H       = SCALAR INTERPOLATION FUNCTIONS                          !12
! LT_FREE = NUMBER OF DEGREES OF FREEDOM                            !13
! LT_GEOM = NUMBER OF GEOMETRY NODES                                !14
! LT_PARM = DIMENSION OF PARAMETRIC SPACE                           !15
! LT_QP   = NUMBER OF QUADRATURE POINTS                             !16
! LT_N    = NUMBER OF NODES PER ELEMENT                             !17
! N_SPACE = DIMENSION OF PHYSICAL SPACE                             !18
! PT      = QUADRATURE COORDINATES                                  !19
! WT      = QUADRATURE WEIGHTS                                      !20
! ...                                                               !21
                                                                    !22
! ..................................................               !23
! ** ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW **          !24
! ..................................................               !25
! Define new local array or variable types, then statements         !26
!                                                                   !27
!      GLOBAL (SINGLE ELEMENT) Galerkin MWR FOR ODE                 !28
!   U,xx + U + X = 0, U(0)=0=U(1), U = Sin(x)/Sin(1) - x            !29
!             With integration by parts                             !30
                                                                    !31
 REAL(DP) :: DL, DX_DR, X_Q  ! Length, Jacobian, Position          !32
 INTEGER  :: IQ              ! Loops                               !33
                                                                    !34
 DL    = COORD (LT_N, 1) - COORD (1, 1) ! LENGTH                   !35
 DX_DR = DL / 2.                        ! CONSTANT JACOBIAN        !36
 S = 0.d0; C = 0.d0                     ! ZERO SUMS                !37
                                                                    !38
 DO IQ = 1, LT_QP        ! LOOP OVER QUADRATURES                    !39
                                                                    !40
!      GET GEOMETRIC INTERPOLATION FUNCTIONS, AND X-COORD           !41
   G   = GET_G_AT_QP (IQ)                       ! parametric       !42
   X_Q = DOT_PRODUCT (G, COORD (1:LT_GEOM, 1)) ! x at point        !43
                                                                    !44
!      GLOBAL INTERPOLATION AND GLOBAL DERIVATIVES (ONLY)           !45
   H   (:)   = (/ (X_Q - X_Q**2), (X_Q**2 - X_Q**3)  /)             !46
   DGH (1,:) = (/ (1 - 2*X_Q),    (2*X_Q - 3*X_Q**2) /)             !47
                                                                    !48
   C = C + H * X_Q * WT (IQ) * DX_DR ! SOURCE, from Q(x)           !49
                                                                    !50
!          SQUARE MATRIX ( SYMMETRIC )                             !51
   S = S + ( MATMUL (TRANSPOSE(DGH), DGH)          & ! from u"      !52
           - OUTER_PRODUCT (H, H)) * WT (IQ) * DX_DR ! from u       !53
                                                                    !54
 END DO  ! QUADRATURE                                               !55
!  Outer product C_sub_jk = A_sub_j * B_sub_k                       !56
!  End of application dependent code                                !57
```

Figure 2.9.1 Global Galerkin with integration by parts

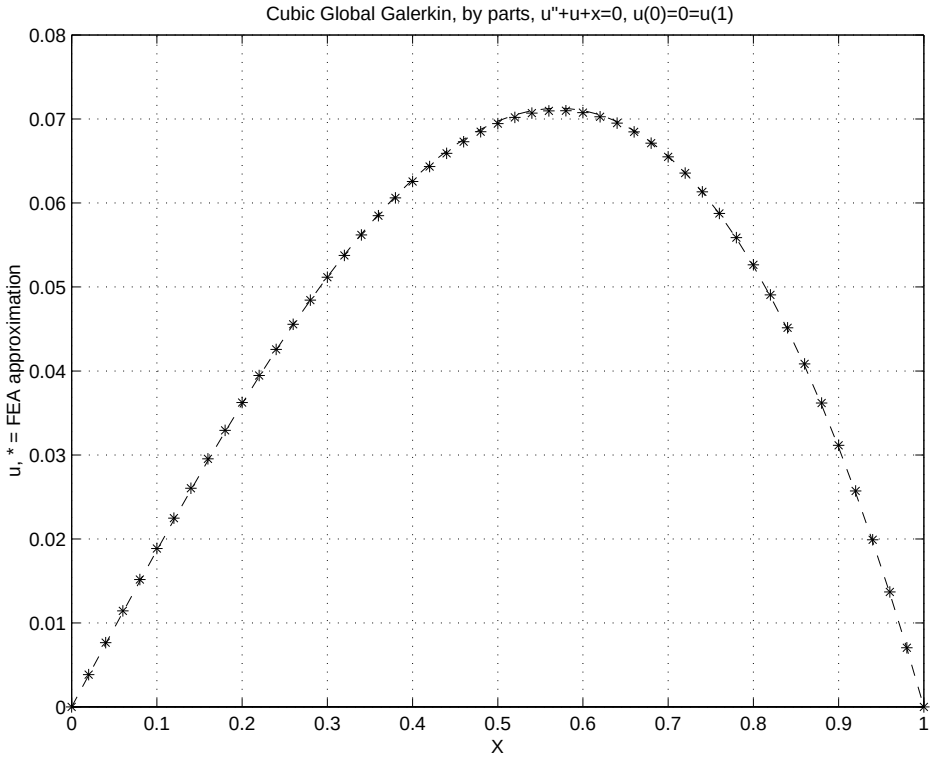

Figure 2.9.2 Exact (-) and cubic global Galerkin (+) solutions

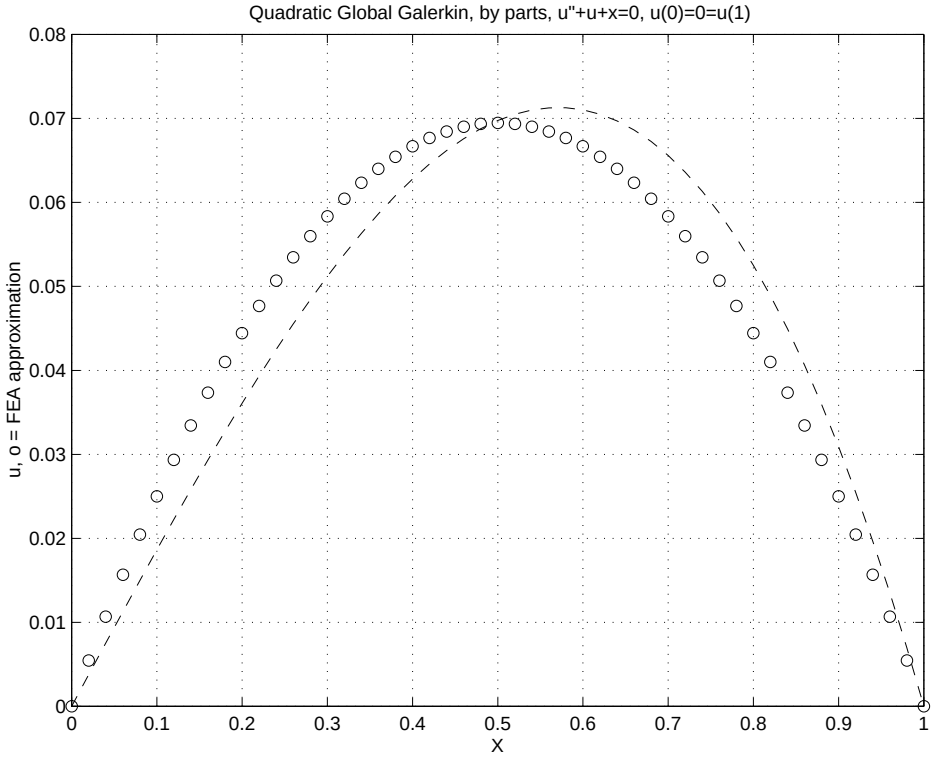

Figure 2.9.3 Exact (-) and quadratic global Galerkin (o) solutions

obtained from the *Galerkin weighted residual method*, followed by integration by parts which introduces the term $du/dx = q$, which we will refer to as the flux. In this case, the Galerkin method states that the function, u, that satisfies the boundary conditions and the integral form:

$$I = \int_0^L \left[(du/dx)^2 - u^2 - u\,Q \right] dx + q_0\,u(0) - q_L\,u(L) = 0, \qquad (2.36)$$

also satisfies Eq. (2.15). For a finite element model we must generate a mesh that subdivides the domain and (usually) its boundary. The unknown coefficients in the finite element model, **D**, will be assigned to the node points of the mesh. Within each element the solution will be approximated by an assumed local spatial behavior. That in turn defines the assumptions for spatial derivatives in an element domain. To illustrate this in one-dimension consider Fig. 2.10.1 which compares an exact solution (dashed) and a piecewise linear finite element model. The domains of influence of a typical element and a typical node are sketched there. In a finite element model, I is assumed to be the sum of the *N_ELEMS* element and *N_BDRY* boundary segment contributions so that

$$I = \sum_{e=1}^{N\_ELEMS} I^e + \sum_{b=1}^{N\_BDRY} I^b, \qquad (2.37)$$

where here $N\_BDRY = 2$ and consists of the last two terms given in Eq. (2.36). A typical element term is

$$I^e = \int_{L^e} \left[(du^e/dx)^2 - (u^e)^2 - Q^e\,u^e \right] dx, \qquad (2.38)$$

where L^e is the length of the element. To evaluate such a typical element contribution, it is necessary to introduce a set of interpolation functions, **H**, so $u^e(x) = \mathbf{H}^e(x)\,\mathbf{D}^e$, and

$$du^e/dx = d\mathbf{H}^e/dx\,\mathbf{D}^e = \mathbf{D}^{e^T}\,d\mathbf{H}^{e^T}/dx,$$

where $\mathbf{D}^e$ denotes the nodal values of u for element e. One of the few standard notations in finite element analysis is to denote the result of the differential operator acting on the interpolation functions, **H**, by the symbol **B**. That is, $\mathbf{B}^e \equiv d\,\mathbf{H}^e/dx$. Thus, a typical element contribution is

$$I^e = \mathbf{D}^{e^T}\,\mathbf{S}^e\,\mathbf{D}^e - \mathbf{D}^{e^T}\,\mathbf{C}^e, \qquad (2.39)$$

where the first contribution to the square matrix is

$$\mathbf{S}^e{}_1 \equiv \int_{L^e} \frac{d\mathbf{H}^{e^T}}{dx}\,\frac{d\mathbf{H}^e}{dx}\,dx = \int_{L^e} \mathbf{B}^{e^T}\,\mathbf{B}^e\,dx,$$

which, for this linear element, has a constant integrand and can be integrated by inspection. The second contribution to the square matrix and the resultant source vector are:

$$\mathbf{S}^e{}_2 \equiv \int_{L^e} \mathbf{H}^{e^T}\,\mathbf{H}^e\,dx, \quad \mathbf{C}^e \equiv \int_{L^e} Q^e\,\mathbf{H}^{e^T}\,dx.$$

Clearly, both the element degrees of freedom, $\mathbf{D}^e$, and the boundary degrees of freedom, $\mathbf{D}^b$, are subsets of the total vector of unknown parameters, **D**. That is, $\mathbf{D}^e \subseteq \mathbf{D}$ and

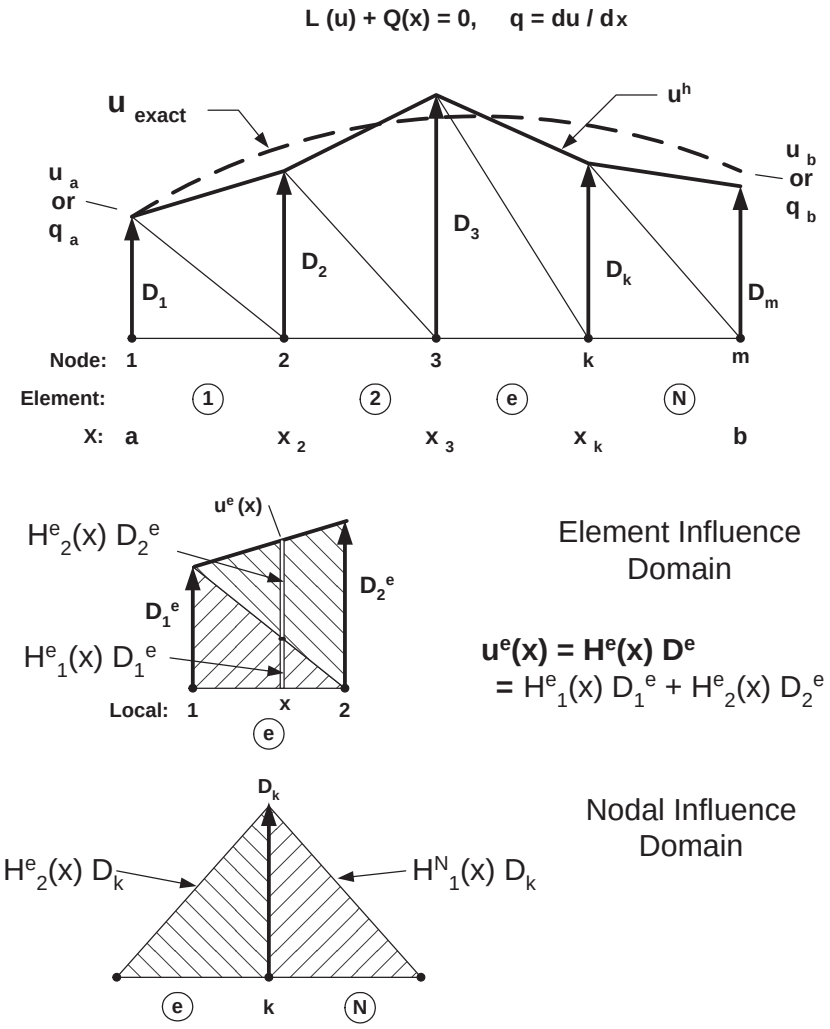

Figure 2.10.1 Finite element influence domains

$\mathbf{D}^b \subset \mathbf{D}$. Of course, the $\mathbf{D}^b$ are usually a subset of the $\mathbf{D}^e$ (i.e., $\mathbf{D}^b \subset \mathbf{D}^e$ and in higher dimensional problems $\mathbf{H}^b \subset \mathbf{H}^e$). The main point here is that $I = I(\mathbf{D})$, and that fact must be considered in the summation. The consideration of the subset relations is merely a bookkeeping problem. This allows Eq. (2.39) to be written as

$$I = \mathbf{D}^T \mathbf{S}\mathbf{D} - \mathbf{D}^T \mathbf{C} = \mathbf{D}^T (\mathbf{S}\mathbf{D} - \mathbf{C}) = 0 \tag{2.40}$$

where

$$\mathbf{S} = \sum_{e=1}^{N\_ELEMS} \boldsymbol{\beta}^{e^T} \mathbf{S}^e \boldsymbol{\beta}^e, \qquad \mathbf{C} = \sum_{e=1}^{N\_ELEMS} \boldsymbol{\beta}^{e^T} \mathbf{C}^e + \sum_{b=1}^{N\_BDRY} \boldsymbol{\beta}^{b^T} \mathbf{C}^b,$$

and where $\boldsymbol{\beta}$ denotes a set of symbolic bookkeeping operations. The combination of the summations and bookkeeping is commonly referred to as the *assembly process*. Note that

one set of operations act on the rows of the column vector and the square matrix while a second set acts on the columns of the square matrix. This is often called "scattering" the element contributions into the corresponding system coefficients.

It is easily shown that for a non-trivial solution, $\mathbf{D} \neq \mathbf{O}$, we must have $\mathbf{O} = \mathbf{S}\,\mathbf{D} - \mathbf{C}$, as the governing algebraic equations to be solved for the unknown nodal parameters, **D**. To be specific, consider a linear interpolation element with two nodes per element, (NOD_PER_EL = 2). If the element length is $L^e = (x_2 - x_1)^e$, then the element interpolation, written in physical coordinates, is

$$\mathbf{H}^e(x) = \left[\frac{({x_2}^e - x)}{L^e} \quad \frac{(x - {x_1}^e)}{L^e} \right], \tag{2.41}$$

so that

$$\mathbf{B}^e = \frac{d\mathbf{H}^e}{dx} = \left[-\frac{1}{L^e} \quad \frac{1}{L^e} \right]. \tag{2.42}$$

Therefore, the two parts to the element square matrix are

$$\mathbf{S}^e{}_1 = \frac{1}{L^e}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{S}^e{}_2 = \frac{L^e}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \tag{2.43}$$

while the element column matrix is

$$\mathbf{C}^e = \int_{L^e} \frac{Q^e}{L^e} \begin{Bmatrix} ({x_2}^e - x) \\ (x - {x_1}^e) \end{Bmatrix} d\,x\,. \tag{2.44}$$

If we were to assume that $Q = Q_0$, a constant, the constant source would simplify to $\mathbf{C}^{e^T} = [1 \;\; 1]\, Q_0\, L^e/2$. That is, the finite element model would replace the constant source per unit length by lumping half its resultant, $Q_0\ L^e$, at each of the two nodes of the element. In the given case of $Q(x) = x$, the source vector reduces to

$$\mathbf{C}^e = \frac{L^e}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} Q_1 \\ Q_2 \end{Bmatrix},$$

where $Q_1 = x_1$ and $Q_2 = x_2$ are the nodal values of the source. If we set $Q_1 = Q_2 = Q_0$ this agrees with the constant source resultant, as noted above.

A typical subroutine segment for implementing this one-dimensional finite element model is shown in Fig. 2.10.2 using a numerically integrated approach. Actually the coding is valid for any line element in the library of interpolation functions (currently linear through cubic) and the selection of element type is set in the control data, as noted later. For a linear element a one point quadrature rule would exactly integrate the first square matrix contribution, but two would be needed for the second contribution and for the column matrix. Clearly higher degree elements require a corresponding increase in the quadrature rule employed. The first 20 lines of that figure relate to an interface that has not yet been described. Only lines 26 and on change with each new application class. Lines 38 and 57 are optional for later post-processing uses. Line 36 accounts for the unit coefficient multiplying the d^2u/dx^2 term in the differential equation. Usually it has some

other assigned user input value.

To compare a finite element spatial approximation with the exact one, and the previous global MWR, select a two element model, as illustrated in Fig. 2.10.3. Let the elements be of equal length, $L^e = L/2$. Here we set $L = 1$ as in the previous global MWR. Then the element square matrices are the same for both elements and forming a common denominator gives the values:

$$\mathbf{S}^e = \frac{1}{24}\begin{bmatrix} 44 & -50 \\ -50 & 44 \end{bmatrix}.$$

The two column matrices will differ because they occupy different regions of space, and thus different sources $Q(x)$. The reader should verify that their numerical values are:

$$\mathbf{C}_{(e=1)}{}^T = \begin{bmatrix} 1 & 2rightfloor/24, \end{bmatrix} \quad \mathbf{C}_{(e=2)}{}^T = \begin{bmatrix} 4 & 5rightfloor/24 \end{bmatrix}$$

Note from Fig. 2.10.3 that these two resultant element vectors account for the total applied source, $Q(x)$, because the sum of the coefficients of the above two vectors is $1/2$ which is the value of the integral of $Q(x)$ over the entire domain.

For the last two terms in Eq. (2.36) note that $u(0) = D_1$ and $u(L) = D_3$ so those terms become $D_1 q_0 - D_3 q_L$. It is not immediately clear that we can write the last two terms as the system level scalar (dot) product $\mathbf{D}^T \mathbf{C}_q$, where the only two non-zero entries in $\mathbf{C}_q$ are q_0 and $-q_L$ (check it out). The assembly process applied to the element matrices and the boundary matrices yields, $\mathbf{S}\,\mathbf{D} = \mathbf{C}$, as

$$\frac{1}{24}\begin{bmatrix} 44 & -50 & 0 \\ -50 & (44+44) & -50 \\ 0 & -50 & 44 \end{bmatrix}\begin{Bmatrix} D_1 \\ D_2 \\ D_3 \end{Bmatrix} = \frac{1}{24}\begin{Bmatrix} 1 \\ (2+4) \\ 5 \end{Bmatrix} - \begin{Bmatrix} q_0 \\ 0 \\ -q_L \end{Bmatrix}.$$

However, these equations do not yet satisfy the two essential boundary conditions of $u(0) = D_1 = u_0 = 0$, and $u(L) = D_3 = u_L = 0$. That is, $\mathbf{S}$ is singular because it is a system of three equations for five unknowns. Note that the essential boundary conditions have assigned values to the two end nodal values (D_1, D_3), so we move their columns (1 and 3) from $\mathbf{S}$ to the right hand side. After applying these conditions, we get

$$\frac{1}{24}\begin{bmatrix} 0 & -50 & 0 \\ 0 & 88 & 0 \\ 0 & -50 & 0 \end{bmatrix}\begin{Bmatrix} D_1 \\ D_2 \\ D_3 \end{Bmatrix} = \frac{1}{24}\begin{Bmatrix} 1 \\ 6 \\ 5 \end{Bmatrix} - \begin{Bmatrix} 24\,q_0 \\ 0 \\ -24\,q_L \end{Bmatrix} - D_1\begin{Bmatrix} 44 \\ -50 \\ 0 \end{Bmatrix} - D_3\begin{Bmatrix} 0 \\ -50 \\ 44 \end{Bmatrix}.$$

Now there are three unknowns (D_2, q_0, q_L) and the system is non-singular. Retaining only the second row, which is the only independent sub-set of equations for a nodal value, and substituting the zero values for D_1, D_3 gives: $88D_2 = 6 + 0 + 0$, or $D_2 = 0.06818$ versus an exact value at that node of $u = 0.06975$.

Now it is possible to return to the remaining unused rows in the algebraic system to recover the flux "reactions" that are necessary to enforce the essential boundary conditions. From the first row


```
! ...  Partial Global Access Arrays                                  ! 1
  REAL(DP) :: C (LT_FREE), S (LT_FREE, LT_FREE) ! Results           ! 2
  REAL(DP) :: PT (LT_PARM, LT_QP), WT (LT_QP)   ! Quadratures       ! 3
  REAL(DP) :: H (LT_N), DGH (N_SPACE, LT_N)     ! Solution & deriv ! 4
  REAL(DP) :: COORD (LT_N, N_SPACE)             ! Elem coordinates ! 5
  REAL(DP) :: XYZ   (N_SPACE)                   ! Pt coordinates   ! 6
                                                                    ! 7
! ...  Partial Notations List                                       ! 8
! COORD     = SPATIAL COORDINATES OF ELEMENT'S NODES                ! 9
! DGH       = GLOBAL DERIVATIVES SCALAR INTERPOLATION FUNCTIONS     !10
! H         = SCALAR INTERPOLATION FUNCTIONS                        !11
! LT_FREE   = NUMBER OF DEGREES OF FREEDOM                          !12
! LT_PARM   = DIMENSION OF PARAMETRIC SPACE                         !13
! LT_QP     = NUMBER OF QUADRATURE POINTS                           !14
! LT_N      = NUMBER OF NODES PER ELEMENT                           !15
! N_SPACE   = DIMENSION OF PHYSICAL SPACE                           !16
! PT        = QUADRATURE COORDINATES                                !17
! WT        = QUADRATURE WEIGHTS                                    !18
! XYZ       = PHYSICAL POINT                                        !19
                                                                    !20
! ..........................................................        !21
!  **  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW **        !22
! ..........................................................        !23
! Define new array or variable types, then give statements          !24
!                                                                   !25
! APPLICATION DEPENDENT Galerkin MWR FOR ODE                        !26
!   U,xx + U + X = 0, with boundary conditions like                 !27
!   U(0)=0=U(1),    so U = Sin(x)/Sin(1) - x         or             !28
!   U(0)=0,U'(1)=0, so U = Sin(x)/Cos(1) - x        etc             !29
                                                                    !30
 REAL(DP) :: DL, DX_DR                  ! Length, Jacobian         !31
 INTEGER  :: IQ                         ! Loops                    !32
                                                                    !33
 DL    = COORD (LT_N, 1) - COORD (1, 1) ! LENGTH                   !34
 DX_DR = DL / 2.                        ! CONSTANT JACOBIAN        !35
 E     = 1.d0                           ! CONSTANT E               !36
                                                                    !37
 CALL STORE_FLUX_POINT_COUNT ! Save LT_QP for post-process          !38
                                                                    !39
 DO IQ = 1, LT_QP        ! LOOP OVER QUADRATURES                    !40
                                                                    !41
!       GET INTERPOLATION FUNCTIONS, AND X-COORD                    !42
   H    = GET_H_AT_QP (IQ)                                         !43
   XYZ  = MATMUL (H, COORD)  ! ISOPARAMETRIC                       !44
                                                                    !45
!       LOCAL AND GLOBAL DERIVATIVES, B = DGH                       !46
   DLH   = GET_DLH_AT_QP (IQ) ; DGH   = DLH / DX_DR                 !47
                                                                    !48
!         SOURCE VECTOR WITH Q(X) = X                               !49
   C = C + H * XYZ (1) * WT (IQ) * DX_DR                            !50
                                                                    !51
!         SQUARE MATRIX                                             !52
   S = S + ( MATMUL (TRANSPOSE(DGH), DGH)                  &        !53
           - OUTER_PRODUCT (H, H)       ) * WT (IQ) * DX_DR         !54
                                                                    !55
!    SAVE FOR FLUX AVERAGING OR POST PROCESSING, B == DGH           !56
   CALL STORE_FLUX_POINT_DATA (XYZ, E, DGH) ! for post-proc         !57
 END DO  ! QUADRATURE                                               !58
!  End of application dependent code                                !59
```

Figure 2.10.2 An element implementation for U" + U + X = 0

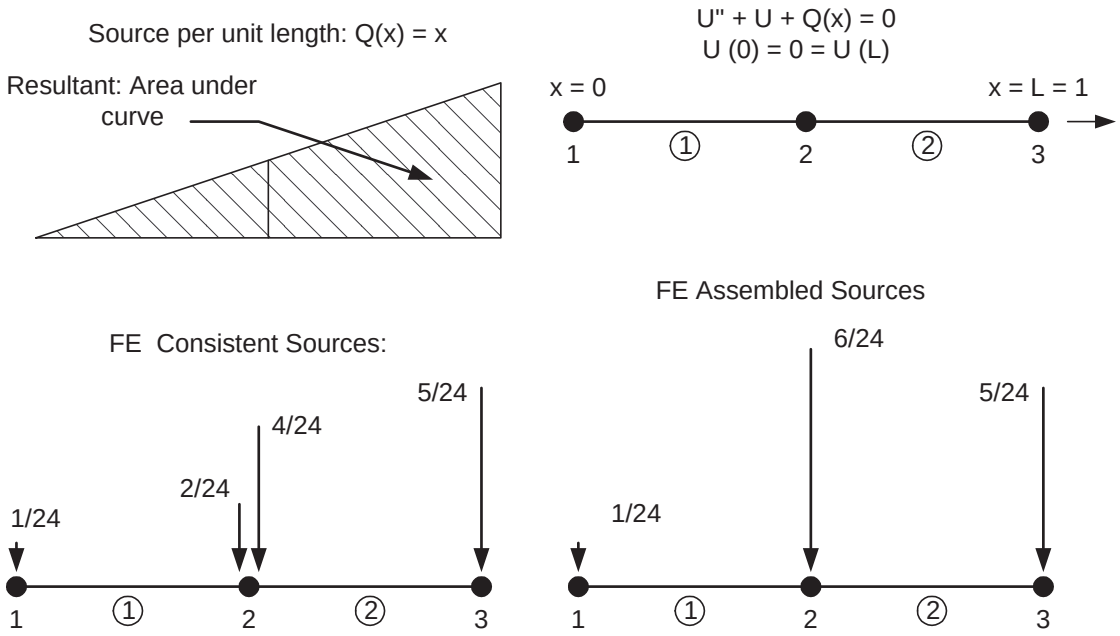

Figure 2.10.3 A two element mesh for $Q(x) = x$

```
title "Two L2 solution of U,xx + U + X = 0" ! 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
gauss       2 ! Maximum number of quadrature points           !10
example   104 ! Application source code library number        !11
exact_case  9 ! Analytic solution for list_exact, etc         !12
pt_list         ! List the answers at each node point         !13
list_exact      ! List given exact answers at nodes, etc      !14
list_exact_flux ! List given exact fluxes at nodes, etc       !15
bar_chart       ! Include bar chart printing in output        !16
post_el         ! Post-process, create n_tape1, for gradients !17
post_2          ! Post-process, create n_tape2, for norm      !18
quit ! keyword input, remarks follow                          !19
1 U,xx + U + X = 0, U(0)=0=U(1), U = Sin(x)/Sin(1) - x        !20
2 Here we use two linear (L2) line elements.                  !21
 1 1 0.            ! node, bc_flag, x                         !22
 2 0 0.5                                                      !23
 3 1 1.00                                                     !24
  1    1    2    ! elem, two nodes                           !25
  2    2    3                                                !26
 1   1 0.        ! node, dof, essential BC value             !27
 7   1 0.        ! end of data                               !28
```

Figure 2.10.4 Data for a two element model

$$0 - 50\,D_2 + 0 \;=\; 1 - 24q_0 - 44D_1 - 0$$

or $-4.4091 = -24q_0$, so that $q_0 = 0.1837$ which compares to the exact flux value of $q_0 = 0.1884$ at $x = 0$. Likewise, the third row of the system yields the reaction $0 - 50\,D_2 + 0 \;=\; 5 + 24q_L + 0 - 44D_3$ so that the second reaction is $q_L = -0.3504$ versus the exact $q_L = -0.3579$ at $x = L$. Note that the reduced equations would allow any values to be assigned to D_1 and D_3 and that the required reaction flux values would change in proportion. Several finite element codes compute the boundary fluxes by computing the gradients in those elements that are adjacent to the boundary where the essential boundary conditions are applied. Getting those fluxes from the integral form, as done above, is usually much more accurate. This will be demonstrated in the post-processing step where we recover the gradients in all the elements in the mesh.

2.11 General Boundary Condition Choices

Our discussion of the model differential equation in Eq. 2.15 has not yet led us to the need to introduce the general range of boundary condition choices that we commonly encounter in applying finite element analysis. Assume our model equation is generalized to the form

$$-\frac{d}{dx}\left(a\frac{du}{dx}\right) + b\,u + c = 0,\;\; 0 < x < L\,.$$

Two physical examples, in one-dimension, readily come to mind where one can assign meanings to the three coefficients in this differential equation. For axial heat transfer u represents the unknown temperature, a the thermal conductivity of the material, b a convection coefficient (per unit length) on the surface, and c a heat source per unit length (and/or information on the external convection temperature). Likewise, for an axial elastic bar u is the displacement, a the material elastic modulus, b a resisting foundation stiffness, and c and axial force per unit length (like gravity).

At either end of the domain one of three conditions can typically be prescribed as possible boundary conditions. The choices are known as a:

1. Dirichlet (essential) boundary condition that specifies the value of the solution on a portion of the boundary; $u(x_D) = U_D$.

2. Neumann (natural) boundary condition that specified the derivative of the solution normal to a portion to the boundary; $\pm\, a\,\partial u\,(x_N)/\partial n = g$. This usually represents a known flux or force, in the direction of the normal vector, defined in terms of the gradient of the solution. The sign of the a coefficient usually depends on the use of a constitutive relation, like Fourier's law, Flick's law, or Hooke's law. In heat transfer it is negative. Note that in one-dimension the direction of the outward normal reverses directions and $\partial/\partial n = \pm\,\partial/\partial x$.

3. Robin (mixed) boundary condition that specifies a linear combination of the normal derivative and solution value on a portion of the boundary; $\pm\, a\,\partial u(x_R)/\partial n + f\,u(x_R) = g$. The mixed condition occurs often in heat transfer as a convection boundary condition and sometimes as a linearized approximation to a radiation condition. In stress analysis it is less common and usually represents the effect of an elastic support foundation, with settlement. The generalized mixed

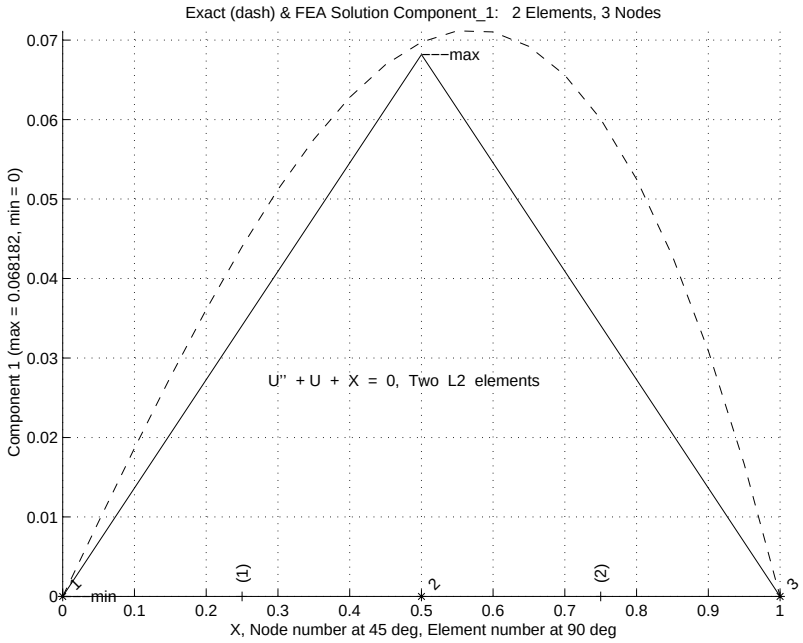

Figure 2.10.5 Results from exact (-) and two linear elements (solid)

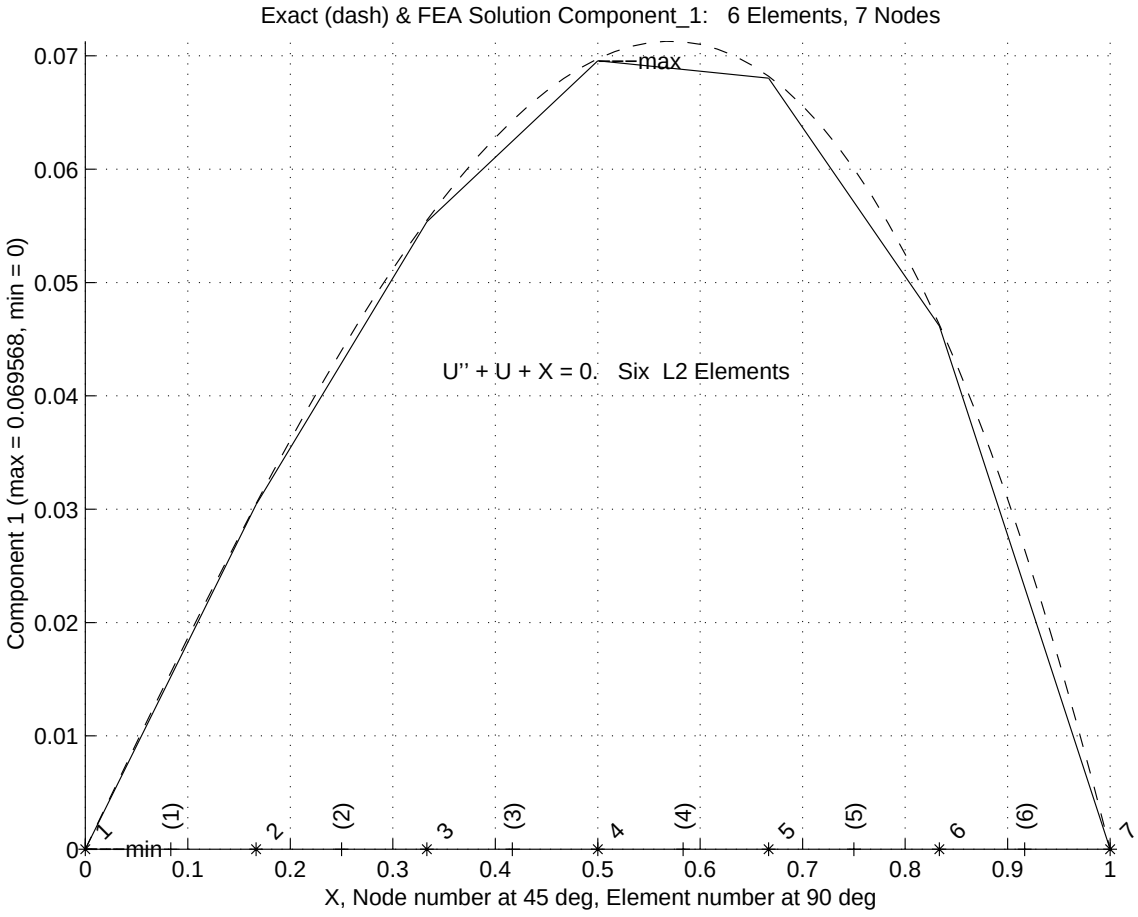

Figure 2.10.6 Results from exact (-) and six linear elements (solid)

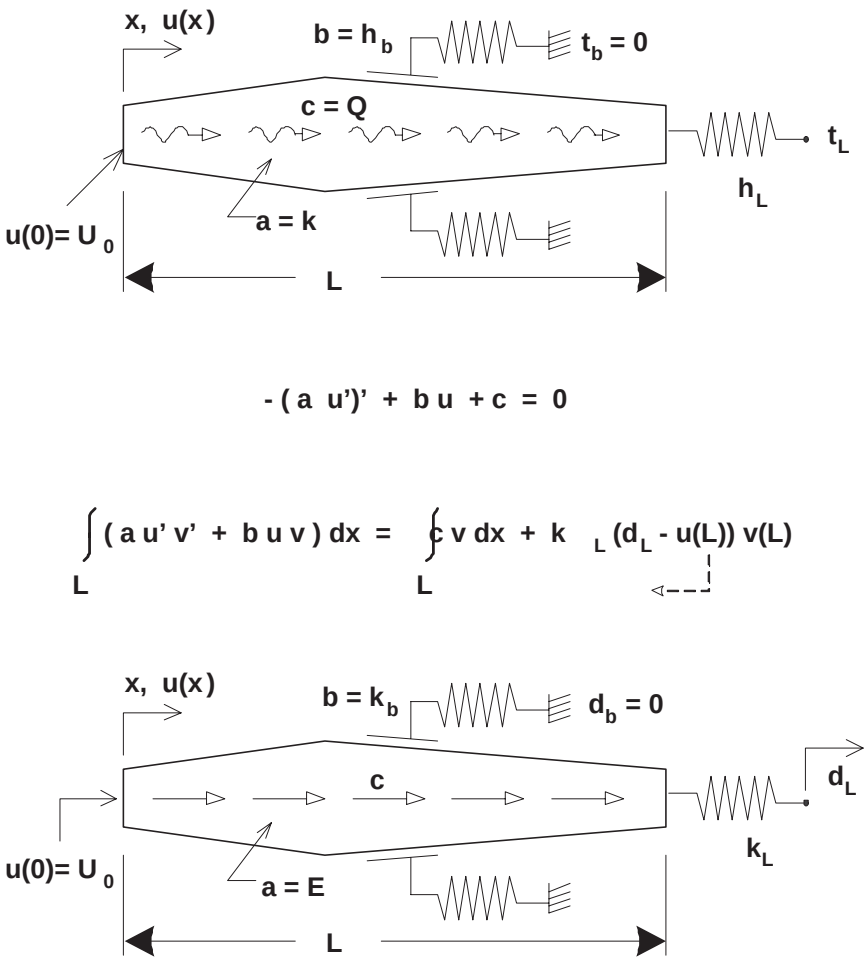

Figure 2.11.1 Examples yielding a mixed boundary condition

condition notation to be used herein is defined as: $a\,\partial u(x_R)/\partial n + r_1\,u(x_R) + r_2 = 0$. For example, on a heat convection boundary one has the requirement $-k(x_R)\,\partial u(x_R)/\partial n = h(x_R)\,[u(x_R) - T_\infty(x_R)]$ so that $r_1 = h$ and $r_2 = -hT_\infty$, where $u(x_R)$ is unknown.

Figure 2.11.1 illustrates stress and thermal problems, in 1-D, with a Dirichlet condition on the left ($x = 0$) end and a Robin condition on the right end ($x = L$).

2.12 General Matrix Partitions

The above small example has lead to the most general form of the algebraic system that results from satisfying the required integral form: a singular matrix system with more unknowns that equations. That is because we chose to apply the essential boundary conditions last and there is not a unique solution until that is done. The algebraic system can be written in a general matrix form that more clearly defines what must be done to reduce the system to a solvable form by utilizing essential boundary condition values. Note that the system degrees of freedom, **D**, and the full equations could always be re-

arranged in the following partitioned matrix form

$$\begin{bmatrix} \mathbf{S}_{uu} & \mathbf{S}_{uk} \\ \mathbf{S}_{ku} & \mathbf{S}_{kk} \end{bmatrix} \begin{Bmatrix} \mathbf{D}_u \\ \mathbf{D}_k \end{Bmatrix} = \begin{Bmatrix} \mathbf{C}_u \\ \mathbf{C}_k + \mathbf{P}_k \end{Bmatrix} \tag{2.45}$$

where $\mathbf{D}_u$ represents the unknown nodal parameters, and $\mathbf{D}_k$ represents the known essential boundary values of the other parameters. The sub-matrices $\mathbf{S}_{uu}$ and $\mathbf{S}_{kk}$ are square, whereas $\mathbf{S}_{uk}$ and $\mathbf{S}_{ku}$ are rectangular, in general. In a finite element formulation all of the coefficients in the **S** and **C** matrices are known. The $\mathbf{P}_k$ term represents that there are usually unknown generalized reactions associated with essential boundary conditions. This means that in general after the essential boundary conditions ($\mathbf{D}_k$) are prescribed the remaining unknowns are $\mathbf{D}_u$ and $\mathbf{P}_k$. Then the net number of unknowns corresponds to the number of equations, but they must be re-arranged before all the remaining unknowns can be computed.

Here, for simplicity, it has been assumed that the equations have been numbered in a manner that places the prescribed parameters (essential boundary conditions) at the end of the system equations. The above matrix relations can be rewritten as

$$\begin{aligned} \mathbf{S}_{uu}\,\mathbf{D}_u + \mathbf{S}_{uk}\,\mathbf{D}_k &= \mathbf{C}_u \\ \mathbf{S}_{ku}\,\mathbf{D}_u + \mathbf{S}_{kk}\,\mathbf{D}_k &= \mathbf{C}_k + \mathbf{P}_k \end{aligned} \tag{2.46}$$

so that the unknown nodal parameters are obtained by inverting the non-singular square matrix $\mathbf{S}_{uu}$ in the top partitioned rows. That is,

$$\mathbf{D}_u = \mathbf{S}_{uu}^{-1}\,(\,\mathbf{C}_u - \mathbf{S}_{uk}\,\mathbf{D}_k\,)\,. \tag{2.47}$$

Most books on numerical analysis assume that you have reduced the system to the non-singular form given above where the essential conditions, $\mathbf{D}_u$, have already been moved to the right hand side. Many authors use examples with null conditions ($\mathbf{D}_k$ is zero) so the solution is the simplest form, $\mathbf{D}_u = \mathbf{S}_{uu}^{-1}\,\mathbf{C}_u$. If desired, the values of the necessary reactions, $\mathbf{P}_k$, can now be determined from

$$\mathbf{P}_k = \mathbf{S}_{ku}\,\mathbf{D}_u + \mathbf{S}_{kk}\,\mathbf{D}_k - \mathbf{C}_k \tag{2.48}$$

In most applications the reaction data have physical meanings that are important in their own right, or useful in validation the solution. However, this part of the calculations is optional. If one formulates a finite element model that satisfies the essential boundary conditions in advance then the second row of the partitioned system **S** matrix is usually not generated and one can not recover reaction data.

Returning to the previous example we post-process it for additional results. We usually initially recover the element gradients at the integration points inside the element. Here we have more than one such point per element (in order to form $\mathbf{S}_2$), even though the approximate gradient is constant in the element. Gathering each element's nodal values back from the solution, $\mathbf{D}^T = [0.\ 0.\ 0.06818\ 0.\ ]$, we compute the fluxes, say $\varepsilon = du/dx$, in the elements from Eq. (2.42) as

$$\boldsymbol{\varepsilon}^e = \frac{1}{L^e} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{Bmatrix} \mathbf{D}_1^e \\ \mathbf{D}_2^e \end{Bmatrix} \tag{2.49}$$

$$\varepsilon^{(1)} = \frac{1}{0.5}\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{Bmatrix} 0.0 \\ 0.06818 \end{Bmatrix} = 0.1364$$

$$\varepsilon^{(2)} = \frac{1}{0.5}\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{Bmatrix} 0.06818 \\ 0.0 \end{Bmatrix} = -0.1364$$

which are the two equal and opposite slopes of the approximate solution. Trying to extrapolate these fluxes from elements at the boundary to the point on the boundary would give much less accurate boundary fluxes than those obtained above from the governing integral form.

The data for the two element model is shown in Fig. 2.10.4. They begin with a group of problem control words and are followed by the numerical data for the nodes, the elements, and the essential boundary conditions. The results from this crude approximation is shown in Fig. 2.10.5 along with the exact solution (as a dashed line). While not exact, the function values are noticed to be most accurate at the nodes. Conversely, the approximate gradients are least accurate at the nodes. The poor function accuracy compared to the previous global MWR solution using three constants is due in part to the fact that two of the three constants have been used to satisfy the boundary conditions and that the prior solution was cubic while the local finite element solution is currently piecewise linear. If we simply increase the number of elements the quality of the approximation will increase as shown in Fig. 2.10.6 where six elements were employed

The previous discussion of the model differential equation showed that to implement a numerical solution we must, as a minimum, code the calculation of an element square matrix, and often also need a column matrix due to a source term. The first six lines of Fig. 2.10.2 hinted that there must be some sort of software interface to a routine that governs such calculations, and that interface provides the storage of the arrays that are generally required for interpolation, integration, position evaluation, etc., and access to any user supplied data. In the MODEL code the routine that is always required is called ELEM_SQ_MATRIX. Figure 2.8.1 summarized other optional and required routines contained within the software library. In order to carry out the above gradient recoveries we either have to recompute the **B** matrix or store it at each quadrature point. That is the purpose of lines 38 and 57 in Fig. 2.10.2. The former declares how many quadrature points are being used by this element type, and later line saves the **B** matrix and the spatial coordinate(s) of the point. (In this example, the "constitutive data" are simply unity and are not really needed.) Thus, the post-processing loop has a similar pair of operations to gather those data and carry out the matrix products used above.

2.13 Elliptic Boundary Value Problems

2.13.1 One-Dimensional Equations

Since the previous model equation had an exact solution that was trigonometric our approximation with polynomials could never give an exact solution (but could reach a specified level of accuracy). Here we will consider an example where the finite element solution can be nodally exact and possibly exact everywhere. The following one-

dimensional model problem which will serve as an analytical example:

$$k\,\frac{d^2t}{dx^2}\;+\;Q\;=\;0 \tag{2.50}$$

on the closed domain, $x \in]\,0, L\,[$, and is subjected to the boundary conditions $t\,(\,L\,) = t_L$, and $k\,dt/dx\;(0) = q_0$. Note that we have dropped the second term in the previous ODE. The corresponding governing integral statement to be used for the finite element model is obtained from the *Galerkin weighted residual method*, followed by integration by parts and is very similar to the first example. In this case, it states that the function, t , which satisfies the essential condition, $t\,(\,L\,) = t_L$, and satisfies

$$I\;=\;\int_0^L\Big[\;k\;(dt/dx)^2 - t\,Q\;\Big]\,dx\;+\;q_0\,t\,(\,0\,)\;-\;q_L\,t\,(\,L\,)\;=\;0\,, \tag{2.51}$$

also satisfies the new ODE. A typical element contribution is (denoting **D** as **T**)

$$I^e\;=\;\mathbf{T}^{e^T}\;\mathbf{S}^e\;\mathbf{T}^e - \mathbf{T}^{e^T}\;\mathbf{C}^e\,, \tag{2.52}$$

where

$$\mathbf{S}^e\;\equiv\;\int_{L^e} k^e\;\frac{d\mathbf{H}^{e^T}}{dx}\;\frac{d\mathbf{H}^e}{dx}\;dx\;=\;\int_{L^e}\mathbf{B}^{e^T}\;k^e\;\mathbf{B}^e\;dx\,,\qquad \mathbf{C}^e\;\equiv\;\int_{L^e} Q^e\;\mathbf{H}^{e^T}\;dx\,.$$

The system of algebraic equations from the weak form is $\mathbf{S}\,\mathbf{T} = \mathbf{C}$ where

$$\mathbf{S}\;=\;\sum_{e\,=\,1}^{N\_ELEMS}\boldsymbol{\beta}^{e^T}\;\mathbf{S}^e\;\boldsymbol{\beta}^e\,,\qquad \mathbf{C}\;=\;\sum_{e\,=\,1}^{N\_ELEMS}\boldsymbol{\beta}^{e^T}\;\mathbf{C}^e\;+\;\sum_{b=1}^{N\_BDRY}\boldsymbol{\beta}^{b^T}\;\mathbf{C}^b\,,$$

and where, as before, $\boldsymbol{\beta}$ denotes a set of symbolic bookkeeping operations. If we select a linear element with the interpolation relations given previously the element matrices are:

$$\mathbf{S}^e\;=\;\frac{k^e}{L^e}\begin{bmatrix}\;1 & -1\;\\ -1 & 1\;\end{bmatrix},\;\mathbf{C}^e\;=\;\int_{L^e}\frac{Q^e}{L^e}\begin{Bmatrix}(x_2{}^e\;-\;x)\\ (x\;-\;x_1{}^e)\end{Bmatrix}\;dx\,.$$

Assuming that $Q = Q_0$, a constant, this simplifies to $\mathbf{C}^{e^T} = [1\quad 1]\,Q_0\,L^e/2$. The exact solution of the original problem for constant $Q = Q_0$ is

$$k\,t(x)\;=\;k\,t_L + q(x\;-\;L) + Q_0\,(\,L^2 - x^2\,)/2\,. \tag{2.53}$$

Since for $Q_0 \neq 0$ the exact value is quadratic and the selected element is linear, our finite element model can give only an approximate solution. However, for the homogeneous problem $Q_0 = 0$, the model can (and does) give an exact solution. To compare a finite element spatial approximation with the exact one, select a two element model. Let the elements be of equal length, $L^e\;=\;L/2$. Then the element matrices are the same for both elements. The assembly process (including boundary effects) yields, $\mathbf{S}\,\mathbf{T} = \mathbf{C}$:

$$\frac{2k}{L}\begin{bmatrix}\;1 & -1 & 0\;\\ -1 & (1+1) & -1\\ 0 & -1 & 1\;\end{bmatrix}\begin{Bmatrix}T_1\\ T_2\\ T_3\end{Bmatrix}\;=\;\frac{Q_0\;L}{4}\begin{Bmatrix}1\\ (1+1)\\ 1\end{Bmatrix}\;-\;\begin{Bmatrix}q_0\\ 0\\ -q_L\end{Bmatrix}.$$

However, these equations do not yet satisfy the essential boundary condition of $t\,(L)\;=\;T_3\;=\;t_L$. That is, **S** is singular and can not be inverted. After applying this

essential boundary condition, the reduced equations are

$$\frac{2k}{L}\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}\begin{Bmatrix} T_1 \\ T_2 \end{Bmatrix} = \frac{Q_2\, L}{4}\begin{Bmatrix} 1 \\ 2 \end{Bmatrix} - \begin{Bmatrix} q \\ 0 \end{Bmatrix} + \frac{2kt_L}{L}\begin{Bmatrix} 0 \\ 1 \end{Bmatrix},$$

or $\mathbf{S}_r\ \mathbf{T}_r\ =\ \mathbf{C}_r$. Solving for $\mathbf{T}_r\ =\ \mathbf{S}_r^{-1}\ \mathbf{C}_r$ yields

$$\mathbf{S}_r^{-1}\ =\ \frac{L}{2k}\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix},\ \ \mathbf{T}_r\ =\ \begin{Bmatrix} T_1 \\ T_2 \end{Bmatrix} = \frac{Q_0\, L^2}{8k}\begin{Bmatrix} 4 \\ 3 \end{Bmatrix} - \frac{qL}{2k}\begin{Bmatrix} 2 \\ 1 \end{Bmatrix} + t_L\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

These are the exact nodal values as can be verified by evaluating the solution at $x = 0$ and $x\ = L/2$, respectively. Thus, our finite element solution is giving an *interpolate* solution. That is, it interpolates the solution exactly at the node points, and is approximate at all other points on the interior of the element. For the homogeneous problem, $Q_0 = 0$, the finite element solution is exact at all points. These properties are common to other finite element problems. The exact and finite element solutions are illustrated in Fig. 2.13.1, where shaded regions show the error in the solution and its gradient. Note that the derivatives are also exact at least at one point in each element. The optimal derivative sampling points will be considered in a later section. For this differential operation, it can be shown that the center point gives a derivative estimate accurate to $0\,(L^{e2})$, while all other points are only $0\,(L^e)$ accurate. For $Q = Q_0$, the center point derivatives are exact in the example.

Next, we want to utilize the last equation from the weighted residual algebraic system to recover the flux "reaction" that is necessary to enforce the essential boundary condition at $x = L$:

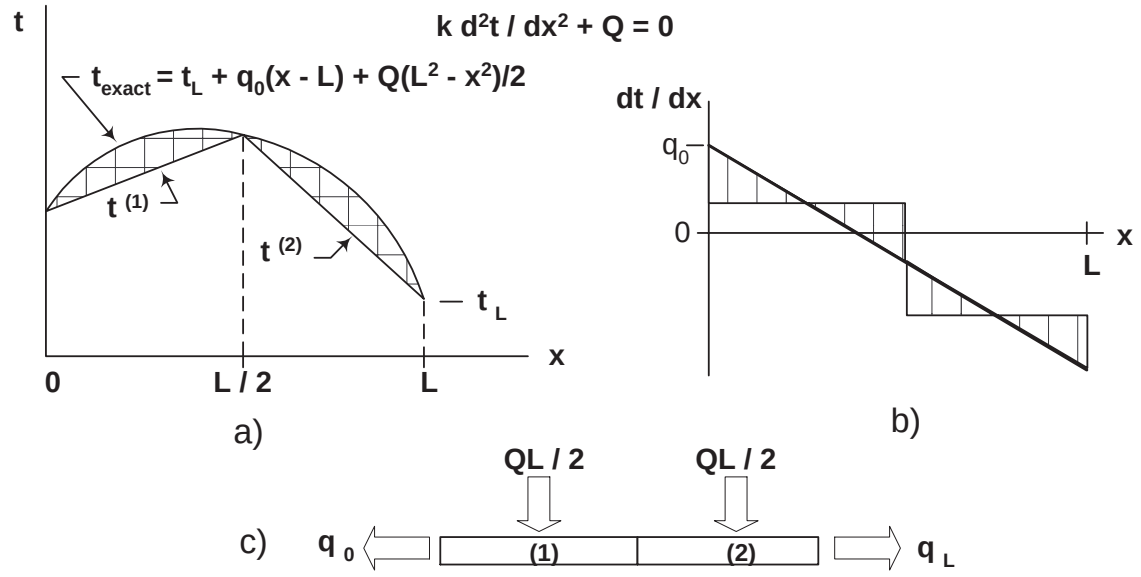

Figure 2.13.1 Example interpolate solution

$$\frac{2k}{L}\left[\ 0 - 1\,T_2 + 1\,T_3\ \right] \ = \ Q_0\,\frac{L}{4} \ - \ (-\,q_L)$$

or $-Q_0 L + q_0 = q_L$, which states the flux equilibrium: that which was generated internally, $Q_0 L$, minus that which exited at $x = 0$, *must* equal that which exits at $x = L$. If one is going to save the reaction data for post-solution recovery, as illustrated above, then one could use programs like SYSTEM_ROW_SAVE and GET_REACTIONS to store and later recover the associated rows of the square matrix.

2.13.2 Two-Dimensional Equations

As an example of the utilization of Galerkin's method in higher dimensions, consider the following model linear operator:

$$L(u) \ = \ \frac{\partial}{\partial x}\left(\ k_x\ \frac{\partial u}{\partial x}\ \right) + \ \frac{\partial}{\partial y}\left(\ k_y\ \frac{\partial u}{\partial y}\ \right) - Q - \zeta\ \frac{\partial u}{\partial t} \ = \ 0\,. \qquad (2.54)$$

Multiplying by a test function, $v(x)$, and integrating $I = \int_\Omega L(u)\,v\,d\,\Omega = 0$. The last two terms are simply

$$I_Q \ = \ \int_\Omega Q\,v\,d\,\Omega\,, \qquad\qquad I_\zeta \ = \ \int_\Omega \zeta\,v\ \frac{\partial u}{\partial t}\ d\,\Omega\,.$$

Our main interest is with the first two terms

$$I_k \ = \ \int_\Omega \left[\ v\ \frac{\partial}{\partial x}\left(\ k_x\ \frac{\partial u}{\partial x}\ \right) + \ v\ \frac{\partial}{\partial y}\left(\ k_y\ \frac{\partial u}{\partial y}\ \right)\right] d\ \Omega$$

which involve the second order partial derivatives. We can remove them by invoking Green's theorem

$$\int_\Omega \left[\ \frac{\partial \Psi}{\partial x} \ - \ \frac{\partial \phi}{\partial y}\ \right] d\ \Omega \ = \ \int_\Gamma \left[\ \phi\ dx + \Psi\ dy\ \right]$$

where we define $\Psi = v\,k_x\,\partial u/\partial x$, and $\phi = -v\,k_y\,\partial u/\partial y$. We note that an alternate form of I_k is

$$I_k \ = \ \int_\Omega \left[\ \frac{\partial}{\partial x}\left(\ v\ k_x\ \frac{\partial u}{\partial x}\ \right) + \ \frac{\partial}{\partial y}\left(\ v\ k_y\ \frac{\partial u}{\partial y}\ \right)\right] d\ \Omega$$

$$- \ \int_\Omega \left[\ \frac{\partial v}{\partial x}\ k_x\ \frac{\partial u}{\partial x} \ + \ \frac{\partial v}{\partial y}\ k_y\ \frac{\partial u}{\partial y}\ \right] d\ \Omega\,.$$

The first term is re-written by Green's theorem

$$I_k = \int_\Gamma \left[-\left(v\, k_y \frac{\partial u}{\partial y} \right) dx + \left(v\, k_x \frac{\partial u}{\partial x} \right) dy \right]$$

$$- \int_\Omega \left[k_x \frac{\partial v}{\partial x} \frac{\partial u}{\partial x} + k_y \frac{\partial v}{\partial y} \frac{\partial u}{\partial y} \right] d\,\Omega\,.$$

For a contour integral with a unit normal vector, **n**, with components [n_x n_y], we note the geometric relations (see Fig. 2.13.2) that $-dx = ds \operatorname{Cos}\Theta_y = ds\, n_y$, and $dy = ds \operatorname{Cos}\Theta_x = ds\, n_x$ which reduces the boundary integral to

$$\int_\Gamma v \left[k_x \frac{\partial u}{\partial x} n_x + k_y \frac{\partial u}{\partial y} n_y \right] ds = \int_\Gamma v\, k_n \frac{\partial u}{\partial n}\, ds$$

where $\partial u / \partial n$ is the normal gradient of u, that is $\nabla u \cdot \mathbf{n}$, and k_n is the value of the orthotropic k in the direction of the normal. The resulting Galerkin form is

$$-I = \int_\Omega \left[k_x \frac{\partial v}{\partial x} \frac{\partial u}{\partial x} + k_y \frac{\partial v}{\partial y} \frac{\partial u}{\partial y} \right] d\,\Omega + \int_\Omega Q\, v\, d\,\Omega$$

$$+ \int_\Omega \zeta\, v \frac{\partial u}{\partial t}\, d\,\Omega - \int_\Gamma v\, k_n \frac{\partial u}{\partial n}\, ds = 0\,. \tag{2.55}$$

Note that Green's theorem brought in the information on the conditions of the surface. If

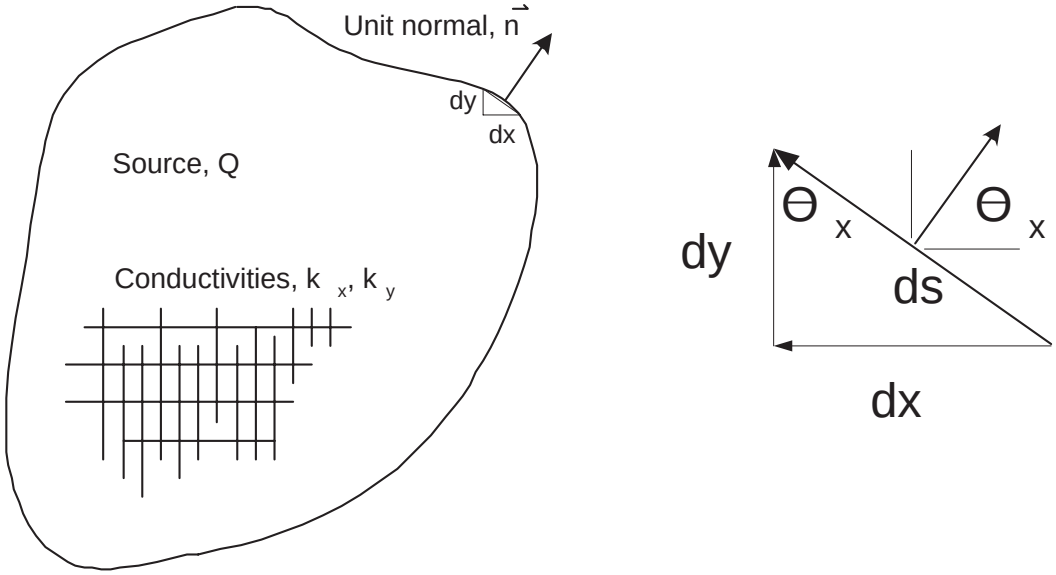

Figure 2.13.2 The unit normal vector components

we split this into element domains so that $\Omega = \underset{e}{\cup} \Omega^e$ and $\Gamma = \underset{b}{\cup} \Gamma^b$, we generate four typical matrices from these terms. We interpolate, as before, with the weights $v(x, y) = \mathbf{H}^e(x, y)\,\mathbf{V}^e$, but include time effects with $u(x, y, t) = \mathbf{H}^e(x, y)\,\mathbf{U}^e(t)$. That is, the unknown coefficients, $\mathbf{U}^e(t)$, are time dependent, and we describe their spatial form in the usual way. The only new concern is that the separation of variables assumption used to represent the time behavior makes the time derivative

$$\frac{\partial u}{\partial t} = \mathbf{H}^e(x, y)\,\frac{\partial}{\partial t}\,\mathbf{U}^e(t) = \mathbf{H}^e(x, y)\,\dot{\mathbf{U}}^e\,. \tag{2.56}$$

These assumptions result in two square matrices

$$\mathbf{S}^e = \int_{\Omega^e} \left[\, \mathbf{H}^{e^T}_{,x}\, k_x\, \mathbf{H}^e_{,x} + \mathbf{H}^{e^T}_{,y}\, k_y\, \mathbf{H}^e_{,y} \,\right] d\,\Omega\,, \quad \mathbf{M}^e = \int_{\Omega^e} \mathbf{H}^{e^T}\, \mathbf{H}^e\, \zeta^e\, d\,\Omega$$

and the two source vectors

$$\mathbf{C}^{\,e}_{Q} = \int_{\Omega^e} \mathbf{H}^{e^T}\, Q^e\, d\,\Omega\,, \quad \mathbf{C}^{\,b}_{q} = \int_{\Gamma^b} \mathbf{H}^{b^T}\, k^b_n\, \frac{\partial u^b}{\partial n}\, d\,\Gamma = \int_{\Gamma^b} \mathbf{H}^{b^T}\, q^b_n\, d\,\Gamma\,,$$

which when assembled yield the system of ordinary differential equations such that $\mathbf{V}^T\,(\,\mathbf{M}\,\dot{\mathbf{U}} + \mathbf{S}\,\mathbf{U} - \mathbf{C}\,) = 0$, so that for arbitrary $\mathbf{V} \neq \mathbf{0}$, we have

$$\mathbf{M}\,\dot{\mathbf{U}} + \mathbf{S}\,\mathbf{U} = \mathbf{C}\,(t)\,, \tag{2.57}$$

which is an initial value problem to be solved from the state at $\mathbf{U}(t = 0)$. We will not consider time dependent problems until later. Here we will note that $\mathbf{M}^e$ and $\mathbf{M}$ are usually called the *element* and *system mass* (or *capacity*) *matrices*, respectively. For the steady state case, $\partial / \partial t = 0$, this reduces to the system of algebraic equations $\mathbf{S}\,\mathbf{U} = \mathbf{C}$ considered earlier. The same procedure is easily carried out in three dimensions. Matrix $\mathbf{S}^e$ changes by having a third term involving z and for $\mathbf{C}^{\,b}_{q}$ we think of $d\,\Gamma$ as a surface area instead of a line segment.

If we had also included a second time derivative term, we could then see by inspection that it would simply introduce another mass matrix (with a different coefficient) times the second time derivative of the nodal degrees of freedom. This would then represent the *wave equation* whose solution in time could be accomplished by techniques to be presented later. If the time derivative terms are not present, and if the term $c\,u$ is included in the PDE and if the coefficient c is an unknown global constant, then this reduces to an *eigen-problem.* That is, we wish to determine a set of *eigenvalues*, c_i, and a corresponding set of *eigenvectors* , or *mode shapes.* It is possible to use completely different assumptions to model time behavior. One increasingly popular approach is the combined *space-time* interpolation which gives a completely different set of element matrices.

2.14 Equivalent Forms *

Analysis problems can be stated in different formats or forms. In finite element methods, we do not deal with the original differential equation (which is called the *strong form*). Instead, we convert to some equivalent integral form. In this optional section, we go through some mathematical manipulations in the one-dimensional case to assure the

reader that they are, indeed, mathematically equivalent approaches to the solution. Here we will consider equivalent forms of certain model differential equations. For most elliptic Boundary Value Problems (*BVP*) we will find that there are three equivalent forms; the original strong form (S), the variational form (V), and the weak form, (W). The latter two integral forms will be reduced to a matrix form, (M). As an example, consider the one-dimensional two-point boundary value problem where we use, $(\,)' = d(\,)/dx$:

$$(S)\quad \left[\begin{array}{lll} & -u''(x) = f(x) & x \in \,]0,1[\\ \text{with} & u(0) = 0 & \\ & u(1) = g\,. & \end{array}\right.$$

That is, given $f : \Omega \to R$ and $g \in R$ find $u : \bar{\Omega} \to R$ such that $u'' + f = 0$ on Ω and $u(0) = 0$, $u(1) = g$. Recall the Hilbert space properties that the function, u, and its first derivative must be square integrable

$$H^1 = \left[\, u\,;\ \int_0^1 [\, u^2 + u_x^2 \,]\ dx < \infty \,\right],$$

and introduce a linear trial solution space with the properties that it satisfies certain boundary conditions, is a Hilbert space, and produces a real number when evaluated on the closure of the solution domain: $S = [\, u\,; u : \bar{\Omega} \to R\,,\ u \in H^1\,,\ u(1) = g\,,\ u(0) = 0\,]$, and a similar linear space of weighting functions with a different set of boundary conditions: $V = [\, \omega\,; \omega : \bar{\Omega} \to R\,,\ \omega \in H^1\,,\ \omega(0) = 0\,,\ \omega(1) = 0\,]$.

The Variational Problem is: given a linear functional, F, that maps S into a real number, $F : S \to R$, given by

$$(V)\quad \left[\begin{array}{l} F(\omega) = \tfrac{1}{2} < \omega'\,,\ \omega' > \ - \ < f\,,\ \omega > \\ \text{Find}\quad u \in S\ \text{ such that }\ F(u) \le F(\omega)\ \text{ for all }\ \omega \in S \end{array}\right.$$

$$(W)\quad \left[\begin{array}{l} \text{The Weak Problem is find}\quad u \in S\ \text{ such that for all }\ \omega \in V \\ \qquad < u'\,,\ \omega' > \ = \ < f\,,\ \omega > \,. \end{array}\right.$$

For this problem, we can show that the strong, variational, and weak form imply the existence of each other: $(S) \Longleftrightarrow (V) \Longleftrightarrow (W)$. In solid mechanics applications (S) would represent the differential equations of equilibrium, (V) would denote the Principal of Minimum Total Potential Energy, and (W) would be the Principal of Virtual Work. Other physical applications have similar interpretations but the three equivalent forms often have no physical meaning. Here, we will introduce the definitions of the common symbols for the *bi-linear form* $a(u,v) \equiv < u'\,,\ v' > \ = \int_0^1 u'v'\,dx$ and the *linear form* $(f,v) = < f\,,v > \ = \int_0^1 f\,v\,dx$.

Here we want to prove the relation that the strong and weak forms imply the existence of each other, $(S) \Longleftrightarrow (W)$. Let u be a solution of (S). Note that $u \in S$ since

$u(1) = g$. Assume that $\omega \in V$. Now set $0 = (u'' + f)$; then we can say

$$0 = -\int_0^1 (u'' + f)\,\omega\,dx = -\int_0^1 u''\,\omega\,dx - \int_0^1 f\,\omega\,dx$$

integrating by parts

$$0 = -u'\omega\Big|_0^1 + \int_0^1 u'\omega'\,dx - \int_0^1 f\,w\,dx\,.$$

Note that since $\omega \in V$ we have $\omega(1) = 0 = \omega(0)$ so that

$$u'\omega\Big|_0^1 = u'(1)\,\omega(1) - u'(0)\,\omega(0) \equiv 0\,,$$

and finally

$$0 = \int_0^1 u'\omega'\,dx - \int_0^1 f\,\omega\,dx\,.$$

Thus, we conclude that $u(x)$ is a solution of the weak problem, (W). That is, $a(u, \omega) = (f, \omega)$ for all $\omega \in V$. Next, we want to verify that the weak solution also implies the existence of the strong form, $(W) \Longleftrightarrow (S)$. Assume that $u(x) \in S$ so that $u(0) = 0$, $u(1) = g$, and assume that $\omega \in V$:

$$\int_0^1 u'\omega'\,dx = \int_0^1 f\,\omega\,dx \qquad \text{for all} \quad \omega \in V\,.$$

Integrating by parts

$$-\int_0^1 (u'' + f)\,\omega\,dx + u'\omega\Big|_0^1 = 0$$

but,

$$u'\omega\Big|_0^1 = u'(1)\,\omega(1) - u'(0)\,\omega(0) \equiv 0\,,$$

since $\omega \in V$, and thus

$$0 = \int_0^1 (u'' + f)\,\omega(x)\,dx\,, \qquad \text{for all} \quad \omega \in V\,.$$

Now we pick $\omega(x) \equiv \phi(x)\,(u'' + f)$ such that ϕ produces a real positive number when evaluated on the domain, $\phi : \overline{\Omega} \to R$, and $\phi(x) > 0$, when $x \in]0, 1[$ and $\phi(0) = \phi(1) = 0$. Is $\omega \in V$? Since $\omega(0) = 0$ and $\omega(1) = 0$, we see that it is and proceed with that substitution,

$$0 = \int_0^1 (u'' + f)^2\,\phi(x)\,dx = \int (\geq 0)\,(> 0)\,dx$$

which implies $(u'' + f) \equiv 0$. Thus, the weak form does imply the strong form, (W) ☞ (S) and combining the above two results we find $(S) \Longleftrightarrow (W)$.

Next, we will consider the proposition that the variational form implies the existence of the weak form, $(V) \Longleftrightarrow (W)$. Assume $u \in S$ is a solution to the weak form (W) and note that $v(1) = \omega(1) + u(1) = 0 + g$. Therefore, $v \in S$. Set $\omega = v - u$ such that

$v = \omega + u$, and $\omega \in V$. Given $F(v) = \frac{1}{2} < v', v' > - < f, v >$ then

$$F(v) = F(u+\omega) = \tfrac{1}{2} < (u'+\omega'),\ (u'+\omega') > - < f,\ u+\omega >$$

$$= \tfrac{1}{2} < u', u' > + < u', \omega' > + \tfrac{1}{2} < \omega', \omega' > .$$

But, since u is a solution to (W) we have $< u', \omega' > - < f, \omega > \equiv 0$, and the above simplifies to $F(v) = F(u) + \frac{1}{2} < \omega', \omega' >$ so that $F(v) \geq F(u)$ for all $v \in S$ since $< \omega', \omega' > \geq 0$. This shows that u is also a solution of the variational form, (V). That is, $(W) \Longleftrightarrow (V)$, which is what we wished to show. Finally, we verify the uniqueness of the weak form, (W). Assume that there are two solutions u_1 and u_2 that are both in the space S. Then $a(u_1, v) = (f, v)$, and $a(u_2, v) = (f, v)$ for all $v \in V$ and subtracting the second from the first $a(u_1 - u_2, v) = 0$, so $< (u_1' - u_2'), v' > = 0$ for all $v \in V$. Consider the choice $v(x) \equiv u_1(x) - u_2(x)$. Is it in V? We note that $v(0) = 0 - 0 = 0$ and $v(1) = g - g = 0$, so we proceed with this choice. Thus, $v' = u_1' - u_2'$ and the inner product is

$$< (u_1' - u_2'),\ (u_1' - u_2') > \ = \ 0 \int_0^1 \left[u_1'(x) - u_2'(x) \right]^2 dx .$$

This means $u_1(x) - u_2(x) = c$. The constant, c, is evaluated from the boundary condition at $x = 0$ so $u_1(0) - u_2(0) = 0 - 0 = c$ which means that $u_1(x) = u_2(x)$, and the weak form solution, (W), is unique.

2.15 Exercises

1. The example ordinary differential equation (ODE)

$$\frac{d^2u}{dx^2} + u + x = 0 \qquad \text{with} \qquad u(0) = 0 = u(1)$$

has the exact solution of $u = sin(x)/sin(1) - x$. Our weighted residual approximations for a global (or single element) solution assumed a cubic polynomial

$$u^*(x) = x(1-x)(c_1 + c_2 x) = h_1(x)c_1 + h_2(x)c_2.$$

The results were (where * denotes a non-unique process):

Method	c_1	c_2
Collocation*	0.1935	0.1843
Least Square	0.1875	0.1695
Galerkin	0.1924	0.1707
Moments*	0.1880	0.1695
Sub-Domain*	0.1880	0.1695

A. Write a program (or spread-sheet) to plot the exact solution and the approximations on the same scale.

B. Modify the above program to plot the error, $u - u^*$, for the Galerkin and Least Square approximations.

C. In the future we will compare solutions by using their norm, $\|u\|^2_{L2} = \int_L u^2 dx$ or $\|u\|^2_{H^1} = \int_L [u^2 + (du/dx)^2] dx$. Compute the $L2$ norms of the exact and Galerkin solutions. Numerical integration is acceptable.

D. Compute the $L2$ norms of the error, $u - u^*$, for the Galerkin and Least Square approximation. Numerical integration is acceptable.

2. Obtain a global Galerkin approximation for the ODE

$$u'' + x^n = 0, \quad x \, \varepsilon \,]\, 0,\, 1[, \quad u(0) = 0 = u(1).$$

Assume a cubic polynomial that satisfies the two boundary conditions:

$$u^*(x) = x(1 - x)\, (\Delta_1 + x\, \Delta_2)\ .$$

For the **S** and **C** matrices. Solve for **Δ** from **SΔ** = **C**. Compare to the exact solution for a) $n = 0$, b) $n = 1$, c) $n = 2$ where

$$u_{exact} = \frac{(x - x^{n+2})}{(n + 1)\, (n + 2)}\ .$$

3. For the one-dimensional problems we often need to evaluate the following three integrals:

$$a)\ \boldsymbol{C}^e = \int_{L^e} \boldsymbol{H}^T dx, \quad b)\ \boldsymbol{M}^e = \int_{L^e} \boldsymbol{H}^T \boldsymbol{H}\, dx,$$

$$c)\ \boldsymbol{S}^e = \int_{L^e} \frac{d\boldsymbol{H}^T}{dx} \frac{d\boldsymbol{H}}{dx}\, dx \quad d)\ \boldsymbol{U}^e = \int_{L^e} \boldsymbol{H}^T \frac{d\boldsymbol{H}}{dx}\, dx$$

which are usually called the resultant source vector, the mass matrix, the stiffness matrix, and the advection matrix, respectively. Analytically integrate these four matrices for the two-noded line element using the physical coordinate "interpolation matrix":

$$\boldsymbol{H}(x) = [H_1(x) \qquad H_2(x)\ ]$$

with $H_1 = (x_2^e - x)/L^e$, $H_2 = (x - x_1^e)/L^e$, $L^e = x_2^e - x_1^e$. Then repeat the four integrals for unit local coordinate interpolations: $H_1(r) = (1 - r)$, $H_2(r) = r$, where we map the coordinate from $0 \le r \le 1$ into $x(r) = x_1^e + L^e r$ using the physical element length of L^e so $dx = L^e dr$ relates the two differential measures, and the mapping yields the physical derivative as $d()/dx = d()/dr\ dr/dx = d()/dr(1/L^e)$. (Which makes the physical units of the results the same as before.) Of course, the two approaches should yield identical algebraic matrix forms.

4. In general, we have a differential operator, $L(u)$ in Ω with essential boundary conditions $u(x) = 0$ on Γ_1 and/or flux boundary conditions

$$q = \frac{\partial u}{\partial n} = \bar{q}(x)\ on\ \Gamma_2$$

where Γ_1 and Γ_2 are non-overlapping parts of the total boundary $\Gamma = \Gamma_1 \cup \Gamma_2$. An approximate solution defines a residual error in Ω, $R(x)$. Errors in the boundary conditions may define two other boundary residual errors: $R_1(x) \equiv u(x) - \bar{u}(x)$, $x\ on\ \Gamma_1$, $R_2(x) \equiv q(x) - \bar{q}$, $x\ on\ \Gamma_2$. Extend our method of weighted residuals to require:

$$\int_\Omega R w d\Omega = \int_{\Gamma_2} (q - \bar{q}) w d\Gamma - \int_{\Gamma_1} (u - \bar{u}) \frac{\partial w}{\partial n} d\Gamma.$$

(Note that these units are consistent.) Assume L(u) is the Laplacian $\nabla^2 u$.

A. Integrating by parts (using Green's Theorem) show that the above form becomes

$$\int_\Omega \frac{\partial u}{\partial x_k} \frac{\partial w}{\partial x_k} d\Omega = \int_{\Gamma_2} \bar{q} w \, d\Gamma + \int_{\Gamma_1} q w \, d\Gamma - \int_{\Gamma_1} \bar{u} \frac{\partial w}{\partial n} d\Gamma + \int_{\Gamma_1} u \frac{\partial w}{\partial n} d\Gamma.$$

(Hint: $\int_\Gamma f \, d\Gamma = \int_{\Gamma_1} f \, d\Gamma + \int_{\Gamma_2} f \, d\Gamma$.) This form is often used in finite element analysis.

B. Integrate by parts again. Show that you obtain

$$\int_\Omega (\nabla^2 w) u \, d\Omega = - \int_{\Gamma_1} \bar{q} w \, d\Gamma - \int_{\Gamma_1} q w \, d\Gamma + \int_{\Gamma_2} u \frac{\partial w}{\partial n} d\Gamma + \int_{\Gamma_1} \bar{u} \frac{\partial w}{\partial n} d\Gamma$$

Picking $w(x)$ such that $\nabla^2 w \equiv 0$ becomes the basis for a boundary element model since only integrals over the boundary remain at that point. We will not consider such methods further.

5. Carry out the integrations necessary to verify the matrix coefficients in **S** and **C** for the model problem in Section 2.5 by:

A. collocation method, B. least squares, C. Galerkin method, D. subdomain method, E. method of moments.

6. Formulate the first order equation $dy/dx + Ay = F$ by A. least squares, B. Galerkin method. Use analytic integration for the linear line element (L2) to form the two element matrices. Compute a solution for $y(0) = 0$ with $A = 2, F = 10$ for 5 uniform elements over $x \leq 0.5$.

2.16 References

[1] Adams, R.A., *Sobolev Spaces*, New York: Academic Press (1975).

[2] Ainsworth, M., "Essential boundary conditions and multi-point constraints in finite element anslysis," *Comp. Meth. Appl. Mech. Eng.*, **190**, pp. 6323−39 (2001).

[3] Axelsson, O. and Baker, V.A., *Finite Element Solution of Boundary Value Problems*, Philadelphia, PA: SIAM (2001).

[4] Aziz, A.K., *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, New York: Academic Press (1972).

[5] Buchanan, G.R., *Finite Element Analysis*, New York: McGraw-Hill (1995).

[6] Ciarlet, P.G., *The Finite Element Method for Elliptical Problems*, Philadelphia, PA: SIAM (2002).

[7] DeBoor, C., ed., *Mathematical Aspects of Finite Elements in Partial Differential Equations*, London: Academic Press (1974).

[8] Heinrich, J.C. and Pepper, D.W., *Intermediate Finite Element Method*, Philadelphia, PA: Taylor & Francis (1999).

[9] Hughes, T.J.R., *The Finite Element Method*, Englewood Cliffs: Prentice-Hall (1987).
[10] Liusternik, L.A. and Sobolev, V.J., *Elements of Functional Analysis*, New York: Frederick Ungar (1961).
[11] Mitchell, A.R. and Wait, R., *The Finite Element Method in Partial Differential Equations*, London: John Wiley (1977).
[12] Nowinski, J.L., *Applications of Functional Analysis in Engineering*, New York: Plenum Press (1981).
[13] Oden, J.T. and Reddy, J.N., *An Introduction to the Mathematical Theory of Finite Elements*, New York: John Wiley (1976).
[14] Oden, J.T., *Applied Functional Analysis*, Englewood Cliffs: Prentice-Hall (1979).
[15] Oden, J.T. and Carey, G.F., *Finite Elements: Mathematical Aspects*, Prentice Hall (1983).
[16] Oden, J.T., "The Best FEM," *Finite Elements in Analysis and Design*, **7**, pp. 103−114 (1990).
[17] Reddy, J.N. and Gartling, D.K., *The Finite Element Method in Heat Transfer and Fluid Dynamics*, Boca Raton, FL: CRC Press (2001).
[18] Strang, W.G. and Fix, G.J., *An Analysis of the Finite Element Method*, Englewood Cliffs: Prentice-Hall (1973).
[19] Szabo, B. and Babuska, I., *Finite Element Analysis*, New York: John Wiley (1991).
[20] Whiteman, J.R., "Some Aspects of the Mathematics of Finite Elements," pp. 25−42 in *The Mathematics of Finite Elements and Applications, Vol. II*, ed. J.R. Whiteman, London: Academic Press (1976).
[21] Zienkiewicz, O.C. and Morgan, K., *Finite Elements and Approximation*, Chichester: John Wiley (1983).
[22] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method*, 4th Edition, New York: McGraw-Hill (1991).

