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 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*:

$$RMS^2 \;=\; \frac{1}{n} \sum_{i=1}^{n} V_i^2 \;=\; \frac{1}{n}\, \mathbf{V}^T \mathbf{V}$$

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. We will introduce various other norms to "measure" these integral quantities.

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

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

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

 © 2004 J.E. Akin

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:

$$\int_a^b p\, dq \;=\; pq\,\Big|_a^b \;-\; \int_a^b q\; dp\,.$$

Error estimator proofs utilize inequalities like the Schwarz inequality

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

and the triangle inequality

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

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.

 © 2004 J.E. Akin. All rights reserved.

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 [15], Nowinski [18], or Oden [20]. This chapter will only cover certain basic topics. Other related advanced works, such as that of Hughes [14], 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 every point. The error at specific points are important and methods for estimating such an error are given by Ainsworth and Oden [2] but will not be considered here. 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 for the space of real numbers, R:

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

An *inner product*, $\langle\hat{}\bullet\,,\bullet\hat{}\rangle$, 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. $< x, x > \geq 0$ and
$< x, x > = 0$ iff $x = 0$
} positive-definiteness,

The pair $x, y \in A$ are said to be orthogonal if $< x, y > = 0$. Another useful property is the Schwarz inequality: $< x, y >^2 \leq < x, x >< y, y >$. 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, $\|\bullet\| : A \to R$, with the properties (for $x, y \in A$ and $\alpha \in R$)

 © 2004 J.E. Akin. All rights reserved.

$$\begin{aligned}
&\text{i.} \quad \left.\begin{aligned} &\| x \| \ \geq \ 0 \quad \text{and} \\ &\| x \| \ = \ 0 \quad \text{iff } x \ = \ 0 \end{aligned}\right\} \ \text{positive-definiteness} \\
&\text{ii.} \quad \| \alpha \ x \| \ = \ | \alpha \ | \ \| x \| \\
&\text{iii.} \quad \| x \ + \ y \| \ \leq \ \| x \| \ + \ \| y \| , \quad \text{triangle inequality} .
\end{aligned} \tag{2.8}$$

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 $(\ ) , _k \ = \partial (\ ) / \partial \ x_k$, 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.

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 a homogeneous differential operator represented as

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

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

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

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

 © 2004 J.E. Akin. All rights reserved.

natural boundary conditions. When $L^* = L$, then $F(u)$ 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 that $< L(u), u > > 0$ 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 *dual problem*. 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

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

Using integration by parts: $\int_a^b p dq = pq\Big|_a^b - \int_a^b q\, dp$. Let $p = v$ so that its derivative is $dp = (dv/dx)\, dx$, 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

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

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

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 $F(v) = 1 * v$ so it applies to the primary variable. The natural boundary condition assigns $G() = G^*() = d()/dx$, which is the gradient or slope of the primary variable. 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.

There are some other general observations about the types of boundary conditions and solution continuity that are associated with even order differential equations. Let the highest order derivative be $2m$. Then the essential boundary conditions involve derivatives of order zero (i.e., the solution itself) through $(m-1)$. The non-essential boundary conditions involve the remaining derivatives of order m through $(2m-1)$. The approximation must maintain continuity of the zero-th through $(m-1)$ derivatives.

 © 2004 J.E. Akin. All rights reserved.

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:

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

with a spatially varying source term $Q(x) = x$, essential boundary conditions of $u = 0$ at $x = 0$ and $u = 0$ at $x = 1$ so that the exact solution to this problem is $u = \text{Sin } x / \text{Sin } 1 - x$. We want to find a global approximate solution involving constants Φ_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

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

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 $g_2\,(x) = x - \text{Sin } x \,/\, \text{Sin } 1$. We could pick $f\,(\,x, \Phi_i\,)$ as a polynomial $f(x) = \Phi_1 + \Phi_2\, x + \cdots \Phi_n\, x^{(n-1)}$. For simplicity, select $n = 2$ and use $g_1\,(x)$ so the approximate solution is

$$u^*\,(x) \;=\; x\,(\,1 \;-\; x\,)\,(\,\Phi_1 + \Phi_2\; x\,) \;=\; \mathbf{h}(x)\,\mathbf{\Phi}\,. \tag{2.17}$$

Expanding, this gives:

$$u^*\,(x) \;=\; (x - x^2)\,\Phi_1 \;+\; (x^2 \;-\; x^3)\,\Phi_2 \;=\; h_1(x)\,\Phi_1 \;+\; h_2(x)\,\Phi_2\,.$$

Here we will employ the MWR to find the $\mathbf{\Phi}$'s. From them we will know the value of $u^*\,(x)$ at all points and compute the error in the solution, $e \;=\; u\,(x) - u^*\,(x)$, and its norm, $\|u\|$. Here, however, we will focus on the residual error in the governing differential equation. From Eqs. 2.15 and 17 we see that the residual error in the differential equation at any point is $R(x) \;=\; u^*\,'' + u^* + Q(x)$, or in expanded form:

$$R(x) \;=\; Q(x) + [\,\frac{d^2}{dx^2} \;+\; 1\,]\,\mathbf{h}(x)\,\mathbf{\Phi}$$

$$R(x) \;=\; Q(x) + [\,\mathbf{h}\,'' + \mathbf{h}\,]\,\mathbf{\Phi} \;=\; Q(x) + \mathbf{b}(x)\,\mathbf{\Phi}$$

$$R(x) \;=\; Q(x) + (\,-2 + x - x^2\,)\,\Phi_1 \;+\; (\,2 - 6x + x^2 - x^3\,)\;\Phi_2 \;\neq\; 0\,. \tag{2.18}$$

where $b_1 \;=\; h_1\,'' + h_1(x) = (0 \;-\; 2) + (x \;-\; x^2)$. For an approximate solution with n constants we can split the residual R into parts including and independent of the Φ_j, say

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

 © 2004 J.E. Akin. All rights reserved.

where **b** is a row matrix and **Φ** 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 :

$$\partial R/\partial \Phi_1 = (-2 + x - x^2), \quad \partial R/\partial \Phi_2 = (2 - 6x + x^2 - x^3),$$

or in general $\partial R/\partial \Phi_j = b_j(x)$. 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 Φ_j. Substituting Eq. 2.19 gives

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

or

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

In matrix form this system of equations is written as:

$$\begin{matrix} [\mathbf{S}] & \{\mathbf{D}\} & = & \{\mathbf{C}\} \\ n \times n & \mathbf{n} \times 1 & & \mathbf{n} \times 1. \end{matrix} \tag{2.22}$$

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

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

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

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

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

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

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

By inspection this reduces Eq. 2.21 to simply

 © 2004 J.E. Akin. All rights reserved.

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

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 Gaussian quadrature points that are used in numerical integration, etc. Pick the two collocation points as $x_1 = 1/4$ and $x_2 = 1/2$; then

$$\begin{bmatrix} \frac{29}{16} & -\frac{35}{64} \\ \frac{7}{4} & \frac{7}{8} \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{4} \\ \frac{1}{2} \end{Bmatrix}$$

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 $\Phi_1 = 6/31$ and $\Phi_2 = 40/217$ so that our first approximate solution is given by $u^* = x(1-x)(42+40x)/217$. 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

$$w_i(x) = \frac{\partial R(x)}{\partial \Phi_i} = b_i(x), \tag{2.25}$$

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

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

Equation 2.26 means in this case Eq. 2.21 becomes

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

For this example

$$\int_0^1 R(x)\,\frac{\partial R}{\partial \Phi_1}\, dx = 0, \quad \int_0^1 R(x)\,\frac{\partial R}{\partial \Phi_2}\, dx = 0$$

and substitutions from Eq. 2.18 gives

 © 2004 J.E. Akin. All rights reserved.

$$\frac{202}{60}\,\Phi_1 \;+\; \frac{101}{60}\,\Phi_2 \;=\; \frac{55}{60}$$

$$\frac{101}{60}\,\Phi_1 \;+\; \frac{393}{105}\,\Phi_2 \;=\; \frac{57}{60}\,.$$

It should be noted from Eqs. 2.19, 21, 25 that this procedure yields a square matrix which is always symmetric. Solving gives $\Phi_1 = 0.188$, $\Phi_2 = 0.170$ 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

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

Here the h_i term defines how we have assumed the contribution from Φ_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

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

and for this specific example we require

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

and Eq. 2.18 yields

$$\frac{3}{10}\,\Phi_1 \;+\; \frac{3}{20}\,\Phi_2 \;=\; \frac{1}{12}$$

$$\frac{3}{20}\,\Phi_1 \;+\; \frac{13}{105}\,\Phi_2 \;=\; \frac{1}{20}$$

which is again symmetric (for the self-adjoint equation). Solving gives degree of freedom values of $\Phi_1 = 71/369$, $\Phi_2 = 7/41$ 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

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

gives the algebraic system

$$\frac{11}{6}\,\Phi_1 \;+\; \frac{11}{12}\,\Phi_2 \;=\; \frac{1}{2}$$

$$\frac{11}{12}\,\Phi_1 \;+\; \frac{19}{20}\,\Phi_2 \;=\; \frac{1}{3}$$

with the solution $\Phi_1 = 122/649$, $\Phi_2 = 110/649$ and selected results at the three interior

 © 2004 J.E. Akin. All rights reserved.

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, Ω_k, 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

$$\begin{bmatrix} \frac{11}{12} & \frac{-53}{192} \\ \frac{11}{12} & \frac{229}{192} \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \end{Bmatrix} = \begin{Bmatrix} \frac{1}{8} \\ \frac{3}{8} \end{Bmatrix}.$$

This results in $\Phi^T = [388 \;\; 352] \,/\, 2068$ and selected results at the three interior points of: 0.043, 0.068, and 0.059, respectively.

These examples show 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 algebraic problem, resulting from an integral relation, for a set of coefficients that define the approximation. 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 examples could have been solved with matrix inversion routines. 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. Practical implementations of direct solvers must account for sparse array storage options and the fact that the factorization operations increases the "fill-in" and thus the total storage requirement.

2.6 Boundary Condition Terms

If a boundary condition involves a non-zero value 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

 © 2004 J.E. Akin. All rights reserved.

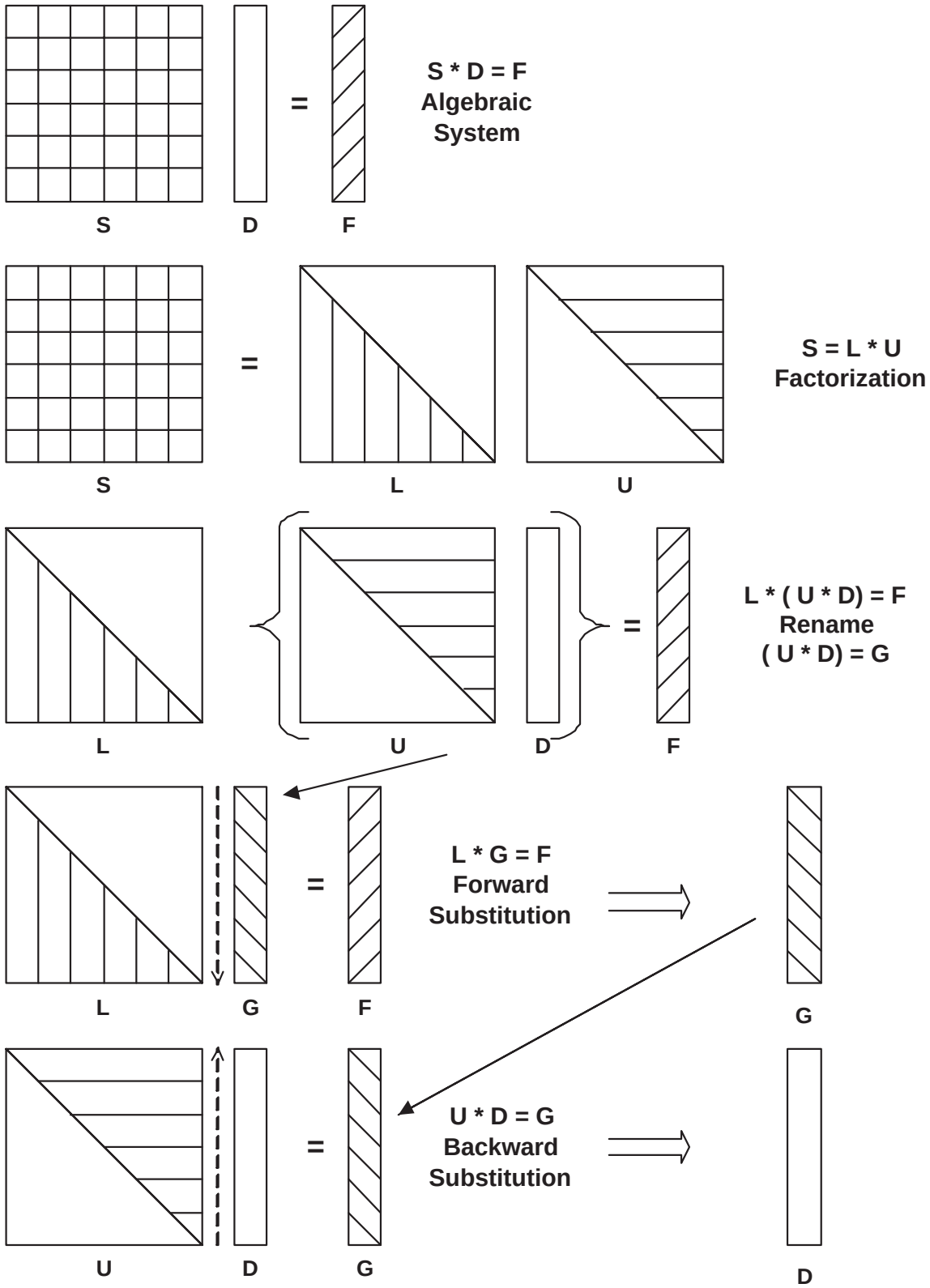


Figure 2.5.1 Steps in the factorization process

 © 2004 J.E. Akin. All rights reserved.

of the corresponding algebraic equations. Of course, since we are going to satisfy both the differential equation and the boundary conditions th total number of algebraic equations developed must be equal to the number of unknown parameters. To illustrate the algebraic procedure that is usually used we will solve the ODE in Eq. 2.15 with an approximation that allows any value to be assigned as the boundary condition at $x = 0$, say $u(0) = \Phi_1$. 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 $x = 0$: $f(x) = \Phi_1 + \Phi_2 x + \Phi_3 x^2$. Then the residual error is

$$R = x + (1 - x)\,\Phi_1 + (x + 2 - x^2)\,\Phi_2 + (x^2 + 2 - 6x - x^3)\,\Phi_3 .$$

Since we now have three unknown degrees of freedom, Φ , 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

$$\begin{bmatrix} -48 & 116 & -35 \\ -32 & 112 & 56 \\ -16 & 116 & 151 \end{bmatrix} \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{Bmatrix} = \begin{Bmatrix} 16 \\ 32 \\ 48 \end{Bmatrix}.$$

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 $u(0) = \Phi_1$. We call this an *essential boundary condition* on Φ_1. There are an infinite number of possible boundary conditions and we gain flexibility by allowing extra constants to satisfy them. Since Φ_1 will be a known number only the last two rows are independent for determining the remaining terms in Φ. 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 Φ :

$$\begin{bmatrix} 112 & 56 \\ 116 & 151 \end{bmatrix} \begin{Bmatrix} \Phi_2 \\ \Phi_3 \end{Bmatrix} = \begin{Bmatrix} 32 \\ 48 \end{Bmatrix} + \begin{Bmatrix} 32 \\ 16 \end{Bmatrix} \Phi_1 .$$

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 $\Phi_1 = 0$ 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 $\Phi_2 = 0.2058$ and $\Phi_3 = 0.1598$. 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 Φ. For example, assume we move the boundary condition to $x = 0.5$. Then evaluating the approximate solution there yields

$$u(0.5) = 0.5\Phi_1 + 0.25\Phi_2 + 0.125\Phi_3$$

which is called a *linear constraint equation* on Φ, or a *multipoint constraint* (MPC). In

 © 2004 J.E. Akin. All rights reserved.

Application Dependent Software

Term / Process	Required (or use keyword example)	Optional
Generate matrices		
Differential operator, Save for flux averages $\mathbf{S}^{E}_{K}$, $\mathbf{M}^{E}$	ELEM_SQ_MATRIX my_el_sq_inc	APPLICATION_B_MATRIX my_b_matrix_inc APPLICATION_E_MATRIX my_e_matrix_inc
Volumetric Source $\mathbf{C}^{E}_{Q}$		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_SQ_MATRIX my_el_sq_inc or ELEM_POST_DATA my_el_post_inc
Post-process element		POST_PROCESS_ELEM my_post_el_inc
Energy norm error estimate	APPLICATION_B_MATRIX APPLICATION_E_MATRIX	my_b_matrix_inc 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*

© 2004 J.E. Akin. All rights reserved.

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, Φ_i , in our test solution. Here we are dealing with polynomials. It is well known that in one-dimension Gaussian quadrature with n_q terms will exactly integrate a polynomial of order $(2\,n_q - 1)$. Gauss proved that this is the minimum number of points that can be used in a summation to 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. 4.4, and Table 4.2.) For example, the Galerkin source term is

$$C_1^e = \int_0^1 x\, h_1(x)\, dx = \sum_{j=1}^{n_q} x_j\, h_1(x_j)\, w_j \tag{2.34}$$

where the x_j and w_j are tabulated data. For $n_q = 2$ 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

 © 2004 J.E. Akin. All rights reserved.

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

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

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

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.

```
          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
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) (IN) X     (MAX_NP, N_SPACE) All nodal coordinates
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
REAL(DP) (OUT) EL_M (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

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.

 © 2004 J.E. Akin. All rights reserved.

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.

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 (see line 52 of Fig. 2.9.1), and we no longer need the storage array for the second derivative of *h* (see line 49 of Figs 2.9.3 & 4). 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 ($\Phi_2 = 0$) this would reduce to a 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. 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 obtained from the *Galerkin weighted residual method*, followed by integration by parts which introduces the term $du/dx = -q$, which we will define as the flux. In higher

 © 2004 J.E. Akin. All rights reserved.

```
! ...  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

 © 2004 J.E. Akin. All rights reserved.

```
! ...  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

 © 2004 J.E. Akin. All rights reserved.

```
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
                                                             !38
! Notes:                                                     !39
!   The "nodal parameters" above do not actually occur       !40
!   at the nodes for a global solution as they will later    !41
!   for all later finite element solutions.                  !42
!                                                            !43
!   There must be as many "nodes" as global degrees of       !44
!   freedom to trick the MODEL code into doing a global      !45
!   solution.  Likewise, there needs to be one fake "element" !46
!   connected to all the nodes.                              !47
                                                             !48
```

Figure 2.8.5 Global MWR solutions and gradients.

 © 2004 J.E. Akin. All rights reserved.

```
! ...  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

 © 2004 J.E. Akin. All rights reserved.

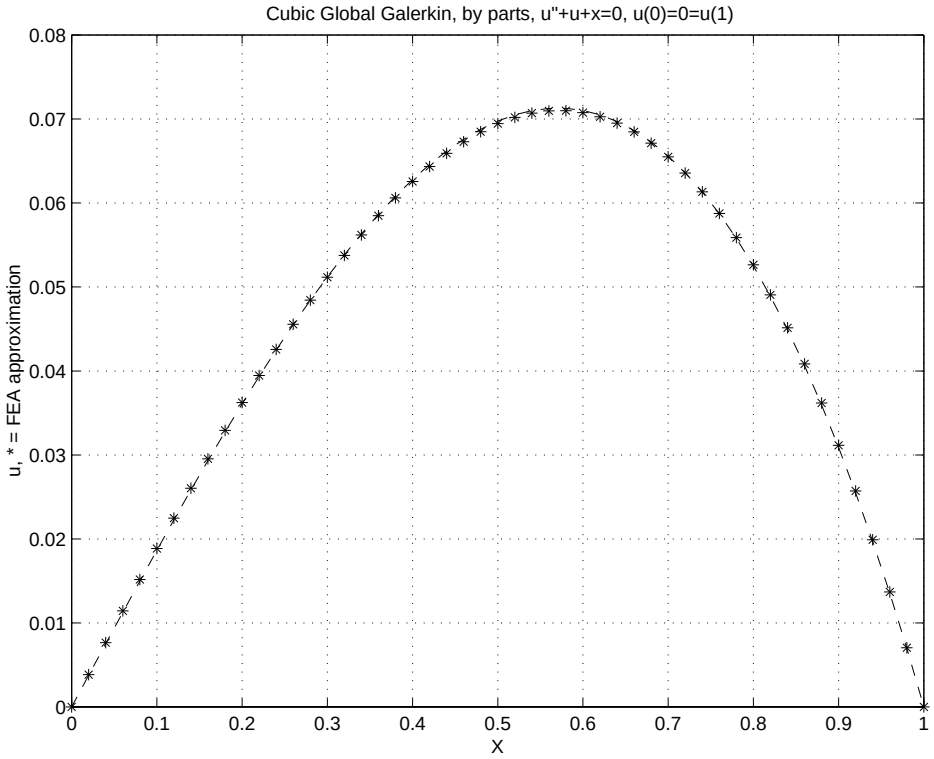


Figure 2.9.2 Exact (-) and cubic global Galerkin (*) solutions

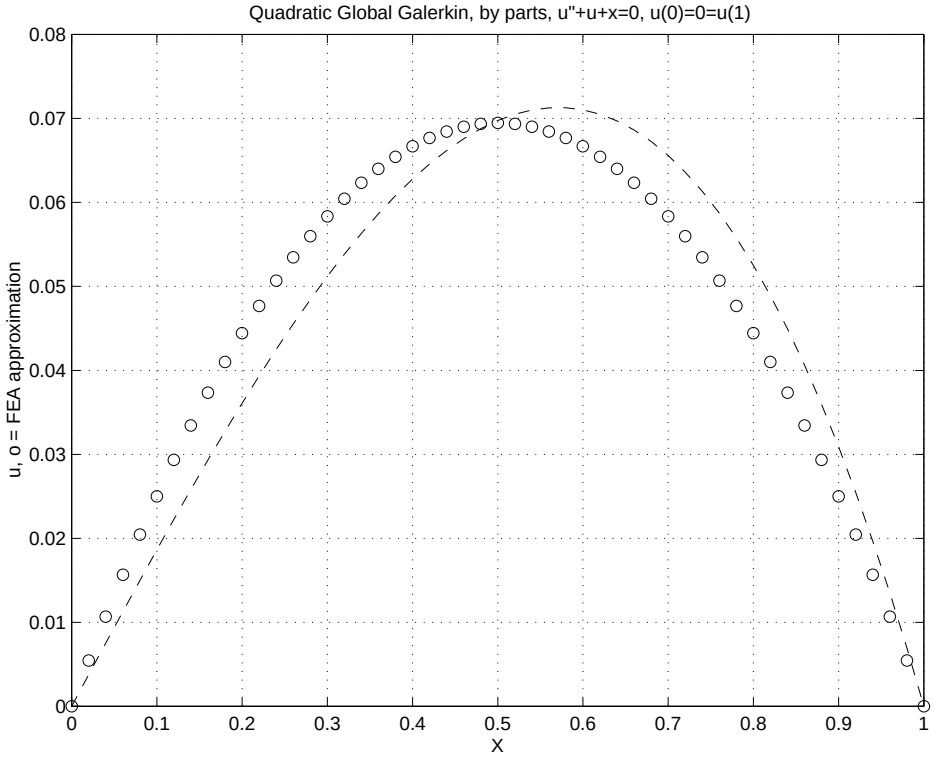


Figure 2.9.3 Exact (-) and quadratic global Galerkin (o) solutions

 © 2004 J.E. Akin. All rights reserved.

dimension problems it is the flux vector, **q**. On a boundary we may be interested in a related scalar term, $q_n = \mathbf{q} \cdot \mathbf{n}$, which is the flux normal to the boundary defined by the unit normal vector **n**. For the special case of the one-dimensional form being considered here we need to note that at the left limit of the domain $\mathbf{n} = -1\,\mathbf{i}$ while at the right limit it is $\mathbf{n} = +1\,\mathbf{i}$. In this case, the Galerkin method states that the function, *v*, that satisfies the boundary conditions and the integral form:

$$I \;=\; \int_0^L \left[\frac{dw}{dx}\,\frac{du}{dx} - w\,u - w\,Q\right] dx \;+\; q_0\,u(0) \;-\; q_L\,u(L) \;=\; 0\,, \qquad (2.36)$$

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_e element and n_b boundary segment contributions so that

$$I \;=\; \sum_{e=1}^{n_e} I^e \;+\; \sum_{b=1}^{n_b} I^b\,, \qquad (2.37)$$

where here $n_b = 2$ and consists of the last two terms given in Eq. 2.36. A typical element term is

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

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

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

with $\mathbf{S}^e = (\mathbf{S}^e{}_1 - \mathbf{S}^e{}_2)$ and 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 square matrix contribution 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 $\mathbf{D}^b \subset \mathbf{D}$. 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

 © 2004 J.E. Akin. All rights reserved.

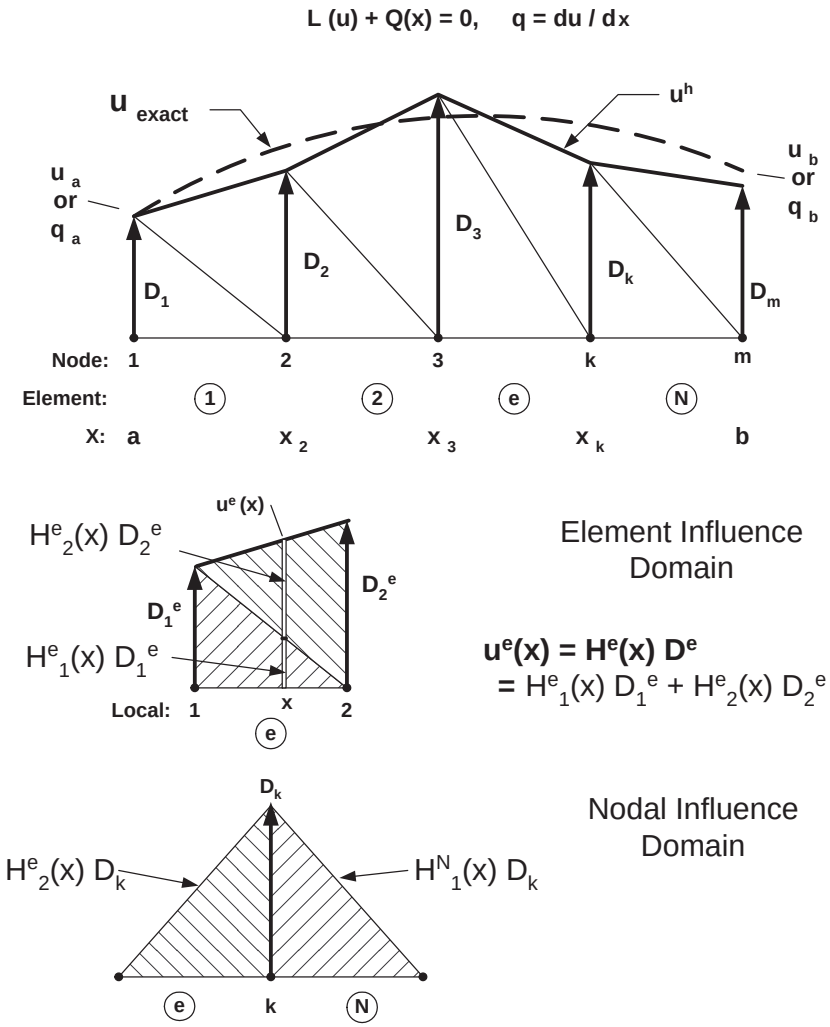


Figure 2.10.1 Finite element influence domains

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

$$\mathbf{S} = \sum_{e=1}^{n_e} \boldsymbol{\beta}^{e^T} \mathbf{S}^e \boldsymbol{\beta}^e , \qquad \mathbf{C} = \sum_{e=1}^{n_e} \boldsymbol{\beta}^{e^T} \mathbf{C}^e + \sum_{b=1}^{n_b} \boldsymbol{\beta}^{b^T} \mathbf{C}^b ,$$

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.

 © 2004 J.E. Akin. All rights reserved.

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, $\mathbf{D}$. To be specific, consider a linear interpolation element with two nodes per element, $(n_n = 2)$. If the element length is $L^e = (x_2 - x_1)^e$, then the element interpolation, written in physical coordinates, is

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

so that

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

Therefore, the two parts (diffusion and convection) to the element square matrix are

$$\mathbf{S}^e{}_1 \;=\; \frac{1}{L^e}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{S}^e{}_2 \;=\; \frac{L^e}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \tag{2.41}$$

while the element column (source) matrix is

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

If we were to assume that $Q = Q_0$, a constant, the constant source would simplify to $\mathbf{C}^{e^T} = \lfloor 1 \;\; 1 \rfloor Q_0\, L^e/2$. 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

$$\mathbf{C}^e \;=\; \frac{L^e}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} Q_1 \\ Q_2 \end{Bmatrix}, \tag{2.42}$$

where $Q_1 = x_1$ and $Q_2 = x_2$ are the nodal values of the source. The latter form is what results if we make the usual assumption that spatially varying data are to be input at the system nodes and interpolated inside the element. In other words, it is common to interpolate from gathered nodal data to define $Q(x) = \mathbf{H}^e(x)\mathbf{Q}^e$ where $\mathbf{Q}^e$ are the local nodal values of the source. Then the resulting integral is the same as in $\mathbf{S}^e{}_2$, so $\mathbf{C}^e = \mathbf{S}^e{}_2\mathbf{Q}^e$ as given above. If we set $Q_1 = Q_2 = Q_0$ this agrees with the constant source resultant, as noted above.

These are all the arrays needed to carry out an analysis if no post-processing information is needed. Thus it is relatively easy to hard-code the source for this model problem. Figure 2.10.2 gives such an implementation as well as including comment statements that look ahead to saving data typically needed for post-processing operations to be introduced later. The element square matrices are defined at lines 18-20 and the matrix multiplication to form $\mathbf{C}^e$ is carried out at line 23, using the nodal values of x which happens in this example to be the nodal values of $Q(x)$. Two optional lines appear as comments at lines 15 and 28. They can be used to save data that can be used later in an error estimate or post-processing. Alternatively, those data could simply be re-computed in a later phase of the program.

 © 2004 J.E. Akin. All rights reserved.

```
!  ..........................................................   ! 1
!   ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***   ! 2
!  ..........................................................   ! 3
!        Hard Coded Galerkin MWR for the ODE                     ! 4
!   E * U,xx + U + x = 0, with E = 1 & boundary conditions like   ! 5
!   U(0)=0=U(1),     so U = sin(x)/sin(1) - x         or          ! 6
!   U(0)=0, U'(1)=0, so U = sin(x)/cos(1) - x        etc         ! 7
                                                                 ! 8
  REAL(DP) :: DL                        ! Length                 ! 9
  REAL(DP) :: S_1 (2, 2), S_2 (2, 2)  ! stiffness, mass          !10
                                                                 !11
  DL = COORD (LT_N, 1) - COORD (1, 1) ! Length                   !12
                                                                 !13
! E  = 1.d0 ; LT_QP = 1       ! constitutive array, quadrature   !14
! CALL STORE_FLUX_POINT_COUNT ! Save LT_QP, for post-processing  !15
                                                                 !16
!          SQUARE MATRIX, CONDUCTION & CONVECTION                !17
  S_1 = RESHAPE ((/ 1, -1, -1, 1 /), (/2,2/)) / DL   ! stiffness !18
  S_2 = DL * RESHAPE ((/ 2,  1,  1, 2 /), (/2,2/)) / 6.d0 ! mass !19
  S   = S_1 - S_2                                        ! net   !20
                                                                 !21
!          INTERNAL SOURCE (EXACT INTEGRATION)                   !22
  C = MATMUL (S_2, COORD (1:2, 1))  ! linear source term         !23
                                                                 !24
!     SAVE FOR FLUX AVERAGING OR POST PROCESSING                 !25
! B (1, :) = (/ -1, 1 /) / DL                  ! dH / dx         !26
! XYZ (1)  = (COORD (LT_N, 1) + COORD (1, 1))/2 ! center point   !27
! CALL STORE_FLUX_POINT_DATA (XYZ, E, DGH)     ! to postprocess  !28
                                                                 !29
! End of application dependent code                              !30
```

Figure 2.10.2 Exact integration linear element model for U" + U + X = 0

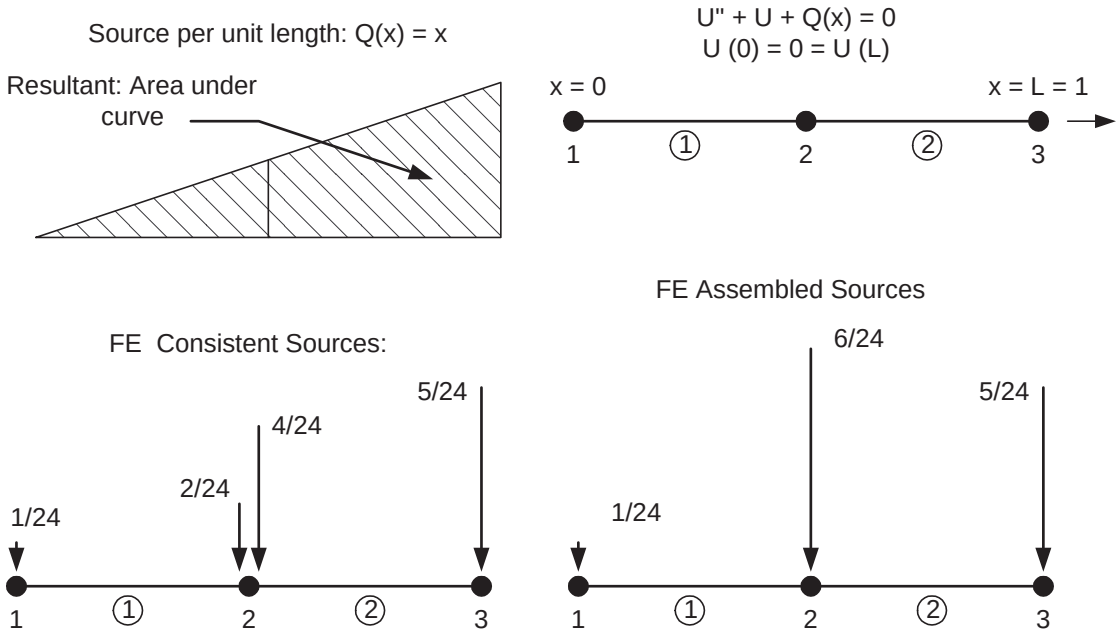


Figure 2.10.3 A two element mesh for $Q(x) = x$

 © 2004 J.E. Akin. All rights reserved.

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:

$$\mathbf{C}_{(e=1)}{}^T = \lfloor 1 \quad 2 \rfloor / 24, \quad \mathbf{C}_{(e=2)}{}^T = \lfloor 4 \quad 5 \rfloor / 24$$

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 $u(0) = \phi_1$ and $u(L) = \phi_3$ so those terms become $\phi_1 q_0 - \phi_3 q_L$. 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

$$\frac{1}{24}\begin{bmatrix} 44 & -50 & 0 \\ -50 & (44+44) & -50 \\ 0 & -50 & 44 \end{bmatrix}\begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{Bmatrix} = \frac{1}{24}\begin{Bmatrix} 1 \\ (2+4) \\ 5 \end{Bmatrix} - \begin{Bmatrix} q_0 \\ 0 \\ -q_L \end{Bmatrix}. \qquad (2.43)$$

However, these equations do not yet satisfy the two essential boundary conditions of $u(0) = \phi_1 = u_0 = 0$, and $u(L) = \phi_3 = u_L = 0$. That is, the above system does not have a unique solution because it is a system of three equations for five unknowns $(\phi_1, \phi_2, \phi_3, q_0, q_L)$. Note that the essential boundary conditions have assigned values to the two end nodal values (ϕ_1, ϕ_3), so we move their columns (1 and 3) from **S** to the right hand side. After applying these conditions and simplifying:

$$\begin{bmatrix} 0 & -50 & 0 \\ 0 & 88 & 0 \\ 0 & -50 & 0 \end{bmatrix}\begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 6 \\ 5 \end{Bmatrix} - \begin{Bmatrix} 24\,q_0 \\ 0 \\ -24\,q_L \end{Bmatrix} - \phi_1\begin{Bmatrix} 44 \\ -50 \\ 0 \end{Bmatrix} - \phi_3\begin{Bmatrix} 0 \\ -50 \\ 44 \end{Bmatrix}.$$

Now there are three unknowns (ϕ_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 ϕ_1, ϕ_3 gives: $88\phi_2 = 6 + 0 + 0$, or $\phi_2 = 0.06818$ versus an exact value at that node of $u = 0.06975$.

Now it is possible to return to the remaining unused rows (1 and 3) in the algebraic system to recover the flux "reactions" that are necessary to enforce the two essential boundary conditions. From the first row

 © 2004 J.E. Akin. All rights reserved.

```
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
bar_chart     ! Include bar chart printing in output         ! 6
exact_case  9 ! Analytic solution for list_exact, etc        ! 7
list_exact    ! List given exact answers at nodes, etc       ! 8
remarks     3 ! Number of user remarks                       ! 9
quit ! keyword input, remarks follow                         !10
1 U,xx + U + X = 0, U(0)=0=U(1), U = sin(x)/sin(1) - x       !11
2 Here we use two linear (L2) line elements.                 !12
3 Defaults to 1-D space, and line element                    !13
 1 1 0.            ! node, bc_flag, x                        !14
 2 0 0.5           ! node, bc_flag, x                        !15
 3 1 1.00          ! node, bc_flag, x                        !16
  1    1    2    ! elem, two nodes                           !17
  2    2    3    ! elem, two nodes                           !18
 1   1 0.        ! node, dof, essential BC value            !19
 3   1 0.        ! end of data                              !20
```

Figure 2.10.4 Data for a two L2 element Galerkin model

```
TITLE: "Two L2 solution of U,xx + U + x = 0"                 ! 1
                                                             ! 2
*** INPUT SOURCE RESULTANTS ***                              ! 3
ITEM       SUM           POSITIVE      NEGATIVE              ! 4
    1    5.0000E-01    5.0000E-01    0.0000E+00              ! 5
                                                             ! 6
*** REACTION RECOVERY ***                                    ! 7
   NODE, PARAMETER,     REACTION, EQUATION                   ! 8
      1,  DOF_1,       1.8371E-01     1                     ! 9
      3,  DOF_1,      -3.5038E-01     3                     !10
                                                             !11
***  RESULTS AND EXACT VALUES IN NODAL ORDER  ***            !12
    NODE,  X-Coord,    DOF_1,      EXACT1,                   !13
       1  0.0000E+00  0.0000E+00  0.0000E+00                 !14
       2  5.0000E-01  6.8182E-02  6.9747E-02                 !15
       3  1.0000E+00  0.0000E+00  0.0000E+00                 !16
```

Figure 2.10.5 Selected two L2 simple Galerkin model results

$$0 - 50\,\phi_2 + 0 \;=\; 1 - 24q_0 - 44\phi_1 - 0$$

or $-4.4091 = -24q_0$, so that $q_0 = 0.1837$ which compares to the exact flux (slope) value of $q_0 = 0.1884$ at $x = 0$. Likewise, the third row of the system yields the reaction $0 - 50\,\phi_2 + 0 \;=\; 5 + 24q_L + 0 - 44\phi_3$ 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 ϕ_1 and ϕ_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 typical post-

 © 2004 J.E. Akin. All rights reserved.

processing steps where we recover the gradients in all the elements in the mesh.

We usually want to recover the element gradients at selected points inside the element. Here we have selected a linear interpolation, so the gradient is constant throughout each element. In such a case we usually report the gradient value at the centroid (center) of the element. Later we will show that location is the most accurate location for the gradient. Gathering each element's nodal values back from the solution, $\mathbf{D}^T = \lfloor 0. \quad 0.06818 \quad 0. \rfloor$, we compute the fluxes, say $\varepsilon = du/dx$, in the elements from Eq. 2.42 as

$$\boldsymbol{\varepsilon}^e = \frac{1}{L^e} \lfloor -1 \qquad 1 \rfloor \begin{Bmatrix} \phi_1^e \\ \phi_2^e \end{Bmatrix} \tag{2.44}$$

$$\varepsilon^{(1)} = \frac{1}{0.5} [-1 \quad 1] \begin{Bmatrix} 0.0 \\ 0.06818 \end{Bmatrix} = 0.1364$$

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

which are the two equal and opposite slopes of this crude 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 (slopes) 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 are listed in Fig. 2.10.5 and shown in Fig. 2.10.6 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. 2.10.7 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

 © 2004 J.E. Akin. All rights reserved.

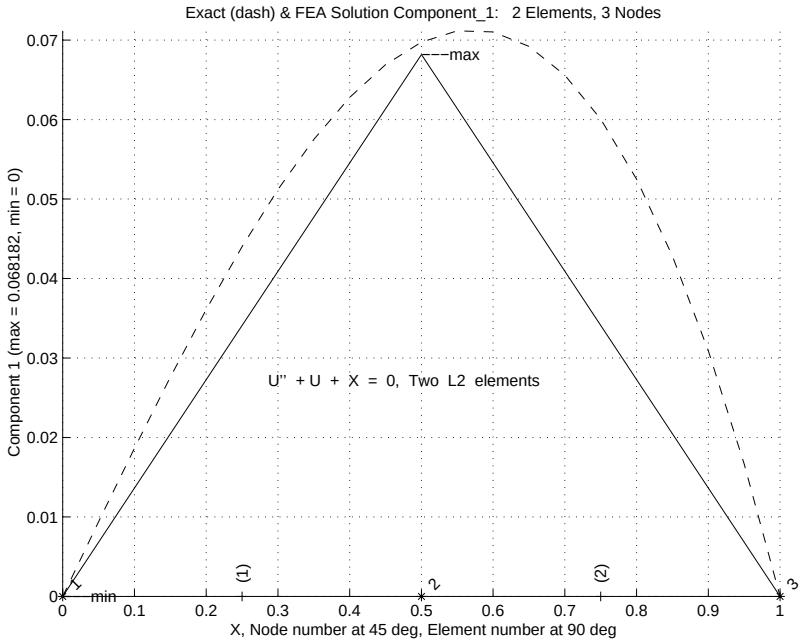


Figure 2.10.6 Results from exact (-) and two linear elements (solid)

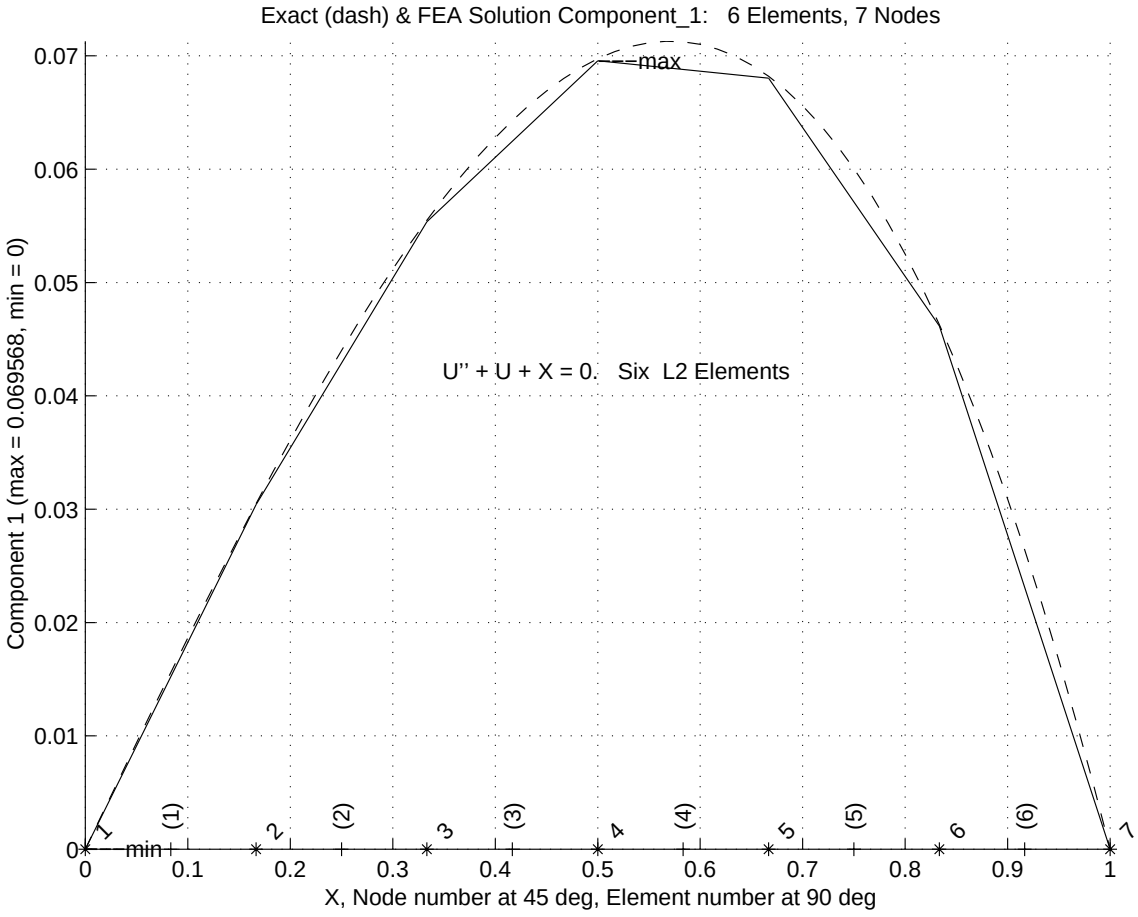


Figure 2.10.7 Results from exact (-) and six linear elements (solid)

 © 2004 J.E. Akin. All rights reserved.

```
! ...  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 via Gauss quadratures          !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 = XYZ (1)                     !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 2.10.8 Element quadrature implementation for U" + U + X = 0

 © 2004 J.E. Akin. All rights reserved.

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.

A typical subroutine segment for implementing any one-dimensional finite element by numerical integration is shown in Fig. 2.10.8. 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 a two point quadrature rule would be needed for the second square matrix 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.

2.11 Continuous Nodal Flux Recovery

Zienkiewicz and Zhu [28, 31, 32] developed the concept of utilizing a local patch of elements, sampled at their super-convergent points, to yield a smooth set of least square fit nodal gradients or fluxes. As noted earlier, the super-convergent points of an element are the special interior locations where the gradients of the element are most accurate. That is, those gradient locations match those of polynomials of one or more degrees higher. Numerous minor improvements to their original process have shown the SCP recovery process to be a practical way to get continuous nodal fluxes, $\boldsymbol{\sigma}^*$. They have demonstrated numerically that one can generate super-convergence estimates for $\boldsymbol{\sigma}^*$ at a node by employing patches of elements surrounding the node. These concepts are illustrated in Fig. 2.11.1.

A local least squares fit is generated over the patch of elements in the following way. Assume a polynomial approximation of the form

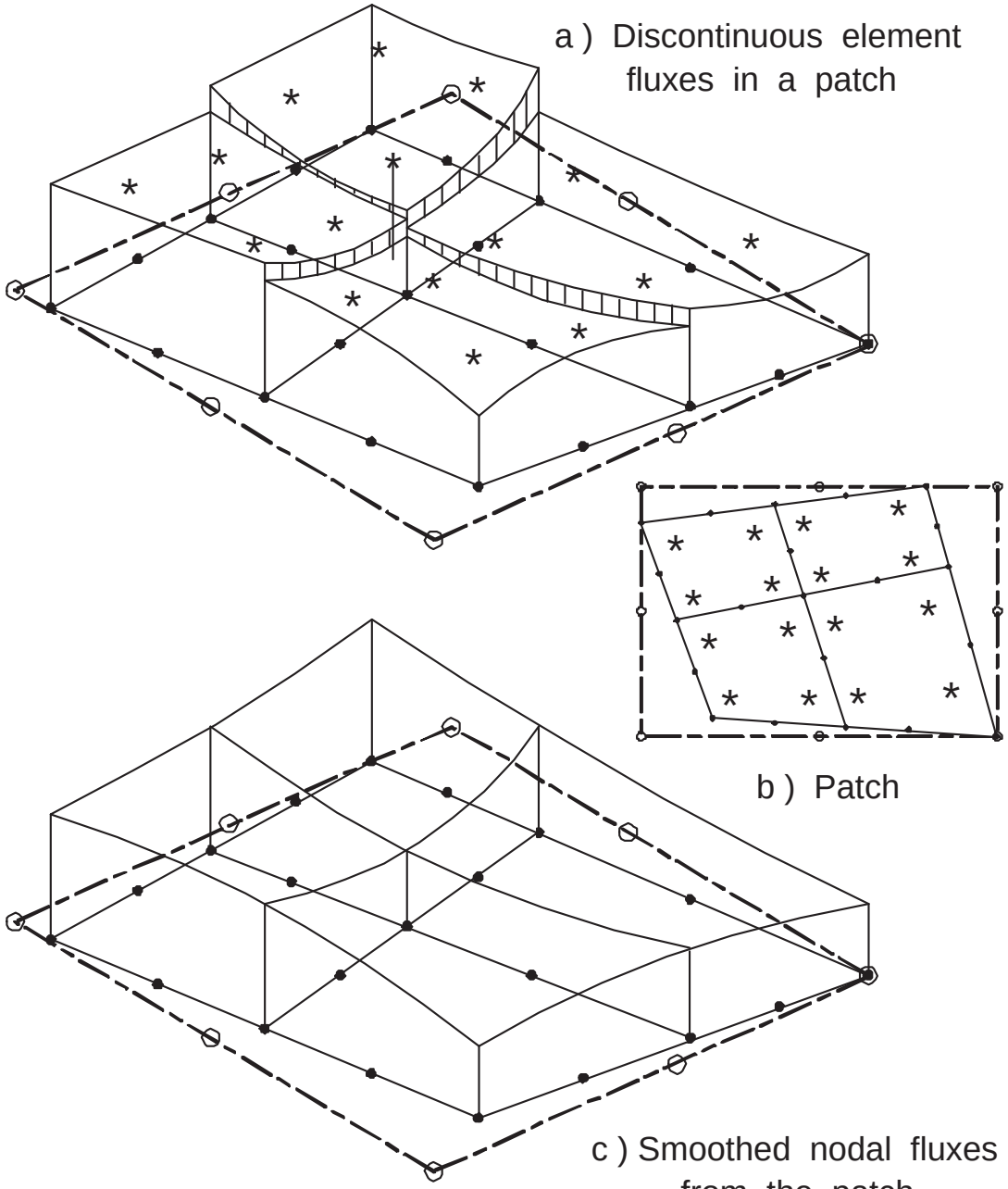
$$\boldsymbol{\sigma}^* = \mathbf{P}(\phi, \eta)\ \mathbf{a} \tag{2.45}$$

where **P** denotes a polynomial (in a local parametric coordinate system selected for each patch) that is of the same degree and completeness that was used to approximate the original solution, $\mathbf{u}_h$. That is, **P** is similar or identical to the solution interpolation array **H**. Here **a** represents a rectangular array that contains the nodal values of the flux. Since $\hat{\boldsymbol{\sigma}}$ was computed using the physical derivatives of **H**, $\hat{\boldsymbol{\sigma}} = \mathbf{E}^e\mathbf{B}^e\boldsymbol{\phi}^e$. To compute the estimate for $\boldsymbol{\sigma}^*$ at the nodes inside the patch, we minimize the function

$$F(\mathbf{a}) = \sum_{j=1}^{n} (\boldsymbol{\sigma}_j^* - \hat{\boldsymbol{\sigma}}_j)^2 \quad \rightarrow \quad \min$$

where n is the total number of integration points (or super-convergent points) used in the elements that define the patch and $\boldsymbol{\sigma}_j$ is the flux evaluated at point $\mathbf{x}_j$. Substituting the two different interpolation functions gives

 © 2004 J.E. Akin. All rights reserved.



Interpolated Solution: $u_h = N(x)\, U$ at node points, •
Element Flux: $q_h = E\, B(x)\, U$ at Gauss points, *
Least Squares Patch Fit of Flux, F_p at patch points, ○
Interpolated Node Flux in Patch: $q_p = N(x)\, F_p$ at nodes in patch, •
Element Flux Error Estimate: $e_q = q_p - q_h$

Figure 2.11.1 Smoothing flux values on a node based patch

 © 2004 J.E. Akin. All rights reserved.

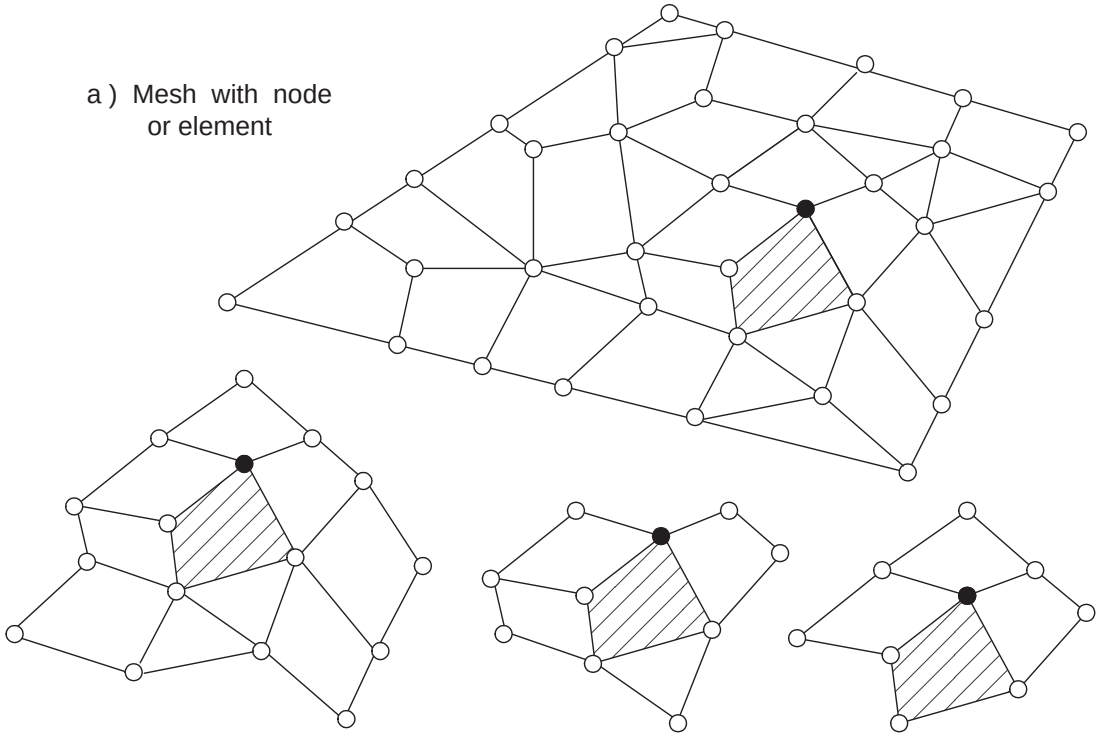


Figure 2.11.2 Examples of element based and node based patches

$$F(\mathbf{a}) = \sum_{e=1}^{n_{pe}} \sum_{j=1}^{n_q} \left[\mathbf{P}_j \mathbf{a} - \mathbf{E}^e \mathbf{B}_j^e \boldsymbol{\phi}^e \right]^2 \tag{2.46}$$

where n_{pe} denotes the number of elements in the patch and n_q is the number of integration points used to form $\hat{\boldsymbol{\sigma}}^e$. That is, we are seeking a least squares fit through the

$$n = \sum_{e=1}^{n_{pe}} \sum_{j=1}^{n_q}$$

data points to compute the unknown coefficients, **a**, which is a rectangular matrix of flux components at each node of the patch. Note that the number rows in the least squares system will be equal to the number of nodes defining the patch "element". Thus the above value of *n* sampling points must be equal to or greater than the number of nodes on the patch "element" (i.e., the number of coefficients in **P**). The standard least squares minimization gives the local algebraic problem $\mathbf{S}\,\mathbf{a} = \mathbf{C}$ where

$$\mathbf{S} = \sum_{e=1}^{n_{pe}} \sum_{j=1}^{n_q} \mathbf{P}^T(\phi_j, \eta_j)\, \mathbf{P}(\phi_j, \eta_j), \quad \mathbf{C} = \sum_{e=1}^{n_{pe}} \sum_{j=1}^{n_q} \mathbf{P}_j^T \mathbf{E}^e \mathbf{B}_j^e \boldsymbol{\Phi}^e . \tag{2.47}$$

This is solved for the coefficients **a** of the local patch fit. It is the cost of solving this small system of equations, for each patch, that we must pay in order to obtain the

 © 2004 J.E. Akin. All rights reserved.

continuous nodal values for the fluxes. To avoid ill-conditioning common to least squares, the local patch fitting parametric space (ϕ , η) is mapped to enclose the patch of elements while using a constant Jacobian for the patch. The use of the constant Jacobian is the key to the efficient conversion of the physical stress location, $\mathbf{x}_j$, to the corresponding patch location, $\boldsymbol{\phi}_j$. Here the implementation actually employs a diagonal constant Jacobian to map the patch onto the physical domain.

Zienkiewicz and Zhu have verified numerically that the derivatives estimated in this way have an accuracy of at least order $O(h^{p+1})$, where h is the size of the element and p is the degree of the interpolation, **N**, used for the solution. There is a theorem that states that if the $\boldsymbol{\sigma}^*$ are super-convergent of order $O(h^{p+\alpha})$ for $\alpha > 0$, then the error estimator will be asymptotically exact. That is, the effectivity index should approach unity, $\Theta \rightarrow 1$. This means that we have the ability to accurately estimate the error and, thus, to get the maximum accuracy for a given number of degrees of freedom. There is not yet a theoretical explanation for the "hyperconvergent" convergence (two orders higher) reported in some of the SCP numerical studies. It may be because the least square fit does not go exactly through the given Barlow points. Thus, they are really sampling nearby. In Sec. 3.8 we showed that derivative sampling points for a cubic are at ± 0.577, while those for the quartic are at ± 0.707. The patch smoothing may effectively be picking up those quartic derivative estimates and jumping to a higher degree of precision.

It is also possible to make other logical choices for selecting the elements that will constitute a patch. Figure 2.11.2 shows two types of element based patches as well as the above node based patch. Regardless of the types of patches selected they almost always overlap with other patches which means that the mesh nodes receive several different estimates for the continuous nodal flux value. They should be quite close to each other but they need to be averaged to get the final values for the continuous nodal fluxes. It is possible to weight that averaging by the size of the contributing patch but it is simpler to just employ a straight numerical average. The implementation of the SCP recovery method will be given in full detail in the next chapter after considering other error indicator techniques.

2.12 A One-Dimensional Example Error Analysis

As a simple example of the process for recovering estimates of the continuous nodal flux values we will return to one of the one-dimensional models considered earlier. Figure 2.12.1 shows a five element model for a second order ODE, while the corresponding analytic, Gauss point (o), and patch averaged flux estimates are shown in Fig. 2.12.2. The piecewise linear flux estimate (solid line) in the latter figure was obtained by using the SCP process described above. It is the relation that will be used to describe $\boldsymbol{\sigma}$ * (**x**) for general post-processing or for use as in the stress error estimate.

For linear interpolation elements we recall that the gradients in each element is constant. The elements used two Gaussian quadrature points (in order to exactly integrate the "mass" matrix). In Figs. 2.11.3 and 4 we see a zoomed view of the various flux representations near element number 2 (in a 5 element mesh). The horizontal dash-dot line through the quadrature point flux values represents the standard finite element spatial distribution, $\mathbf{B}^e \boldsymbol{\phi}^e$, of the flux in that element. Again, the solid line is the spatial form of an averaged flux from a set of patches, and the dashed line is the exact flux

 © 2004 J.E. Akin. All rights reserved.

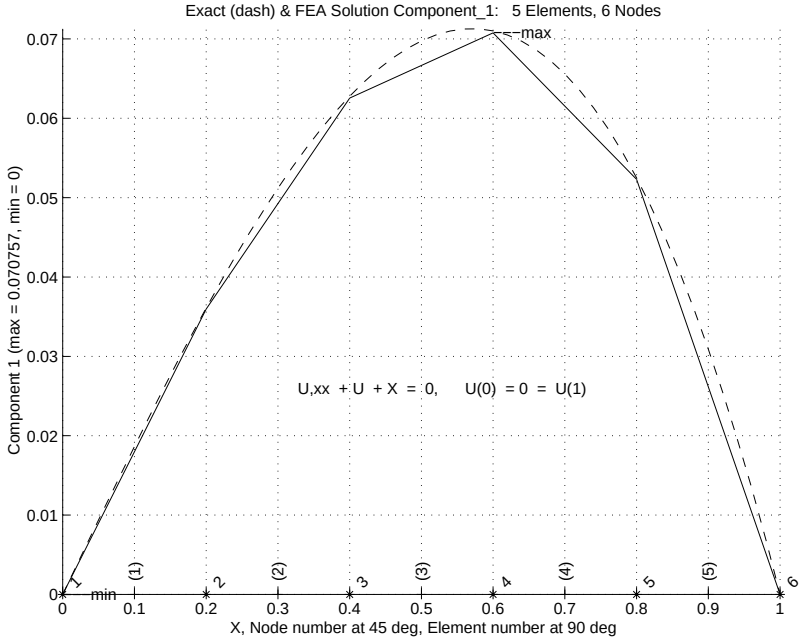


Figure 2.12.1 Analytic and linear element solution results

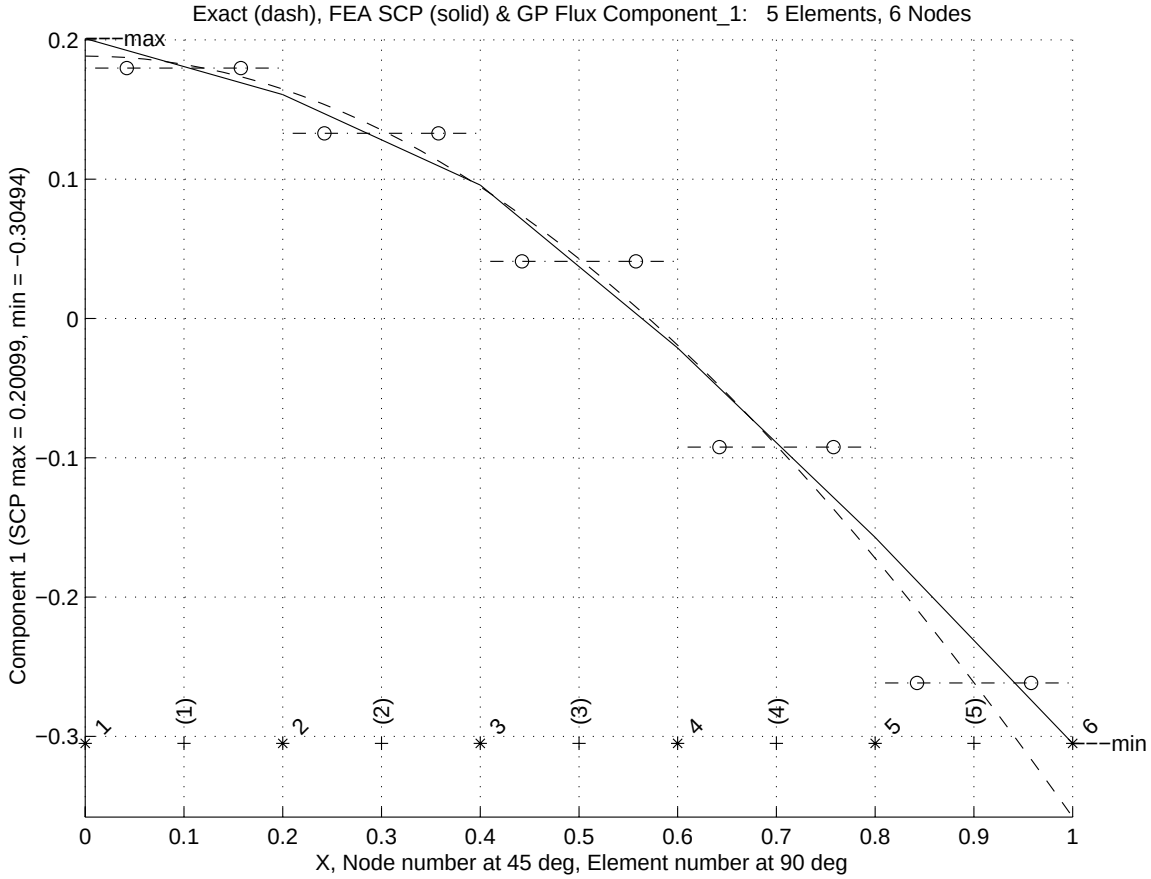


Figure 2.12.2 Exact, patch averaged, and Gauss point flux distribution

 © 2004 J.E. Akin. All rights reserved.

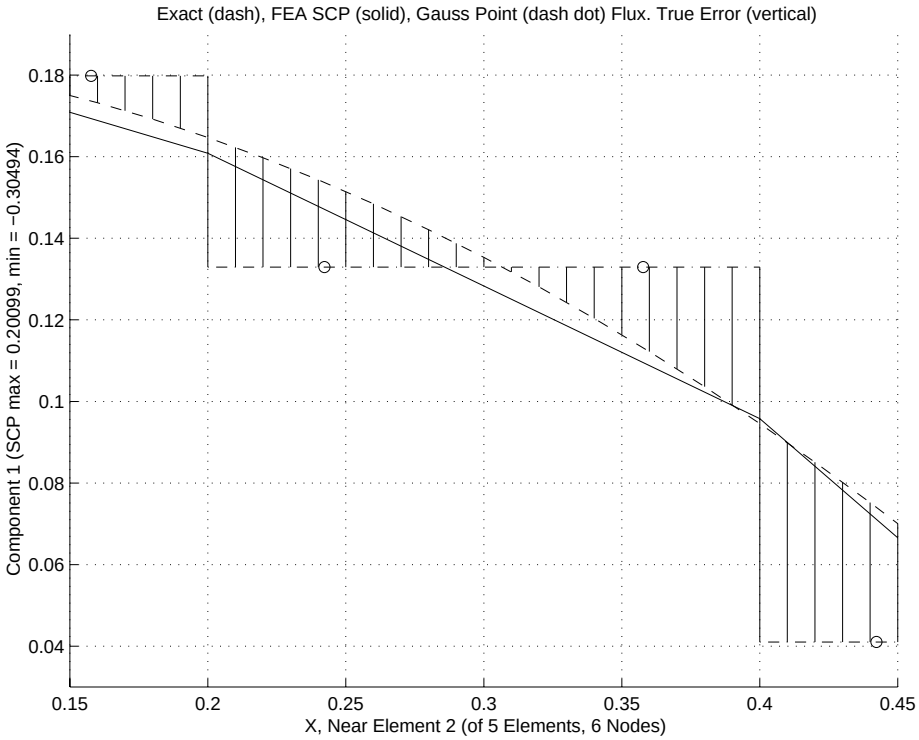


Figure 2.12.3 Zoom of exact flux error near the second element

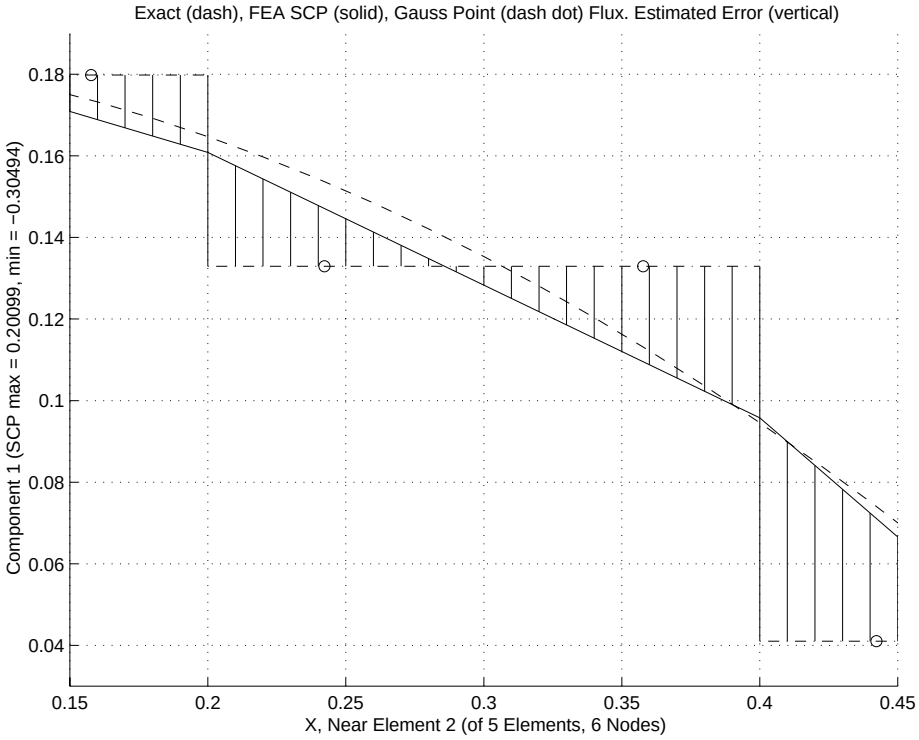


Figure 2.12.4 Zoom of estimated flux error near the second element

 © 2004 J.E. Akin. All rights reserved.

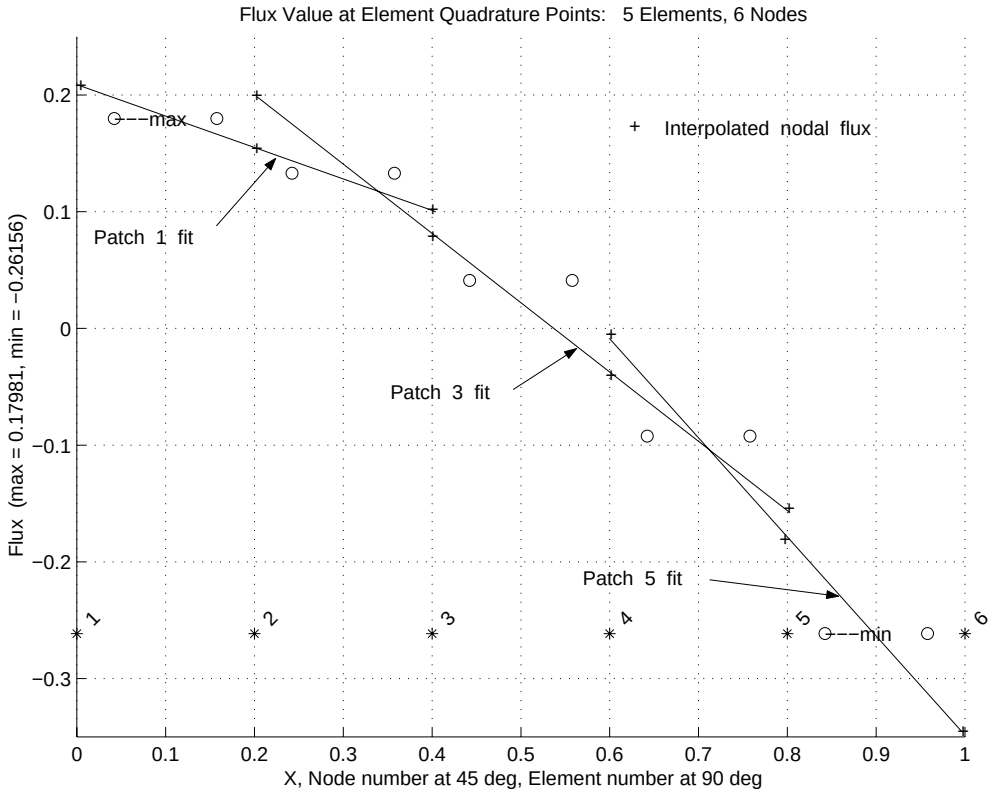


Figure 2.12.5 Element flux and linear patch fits (odd elements)

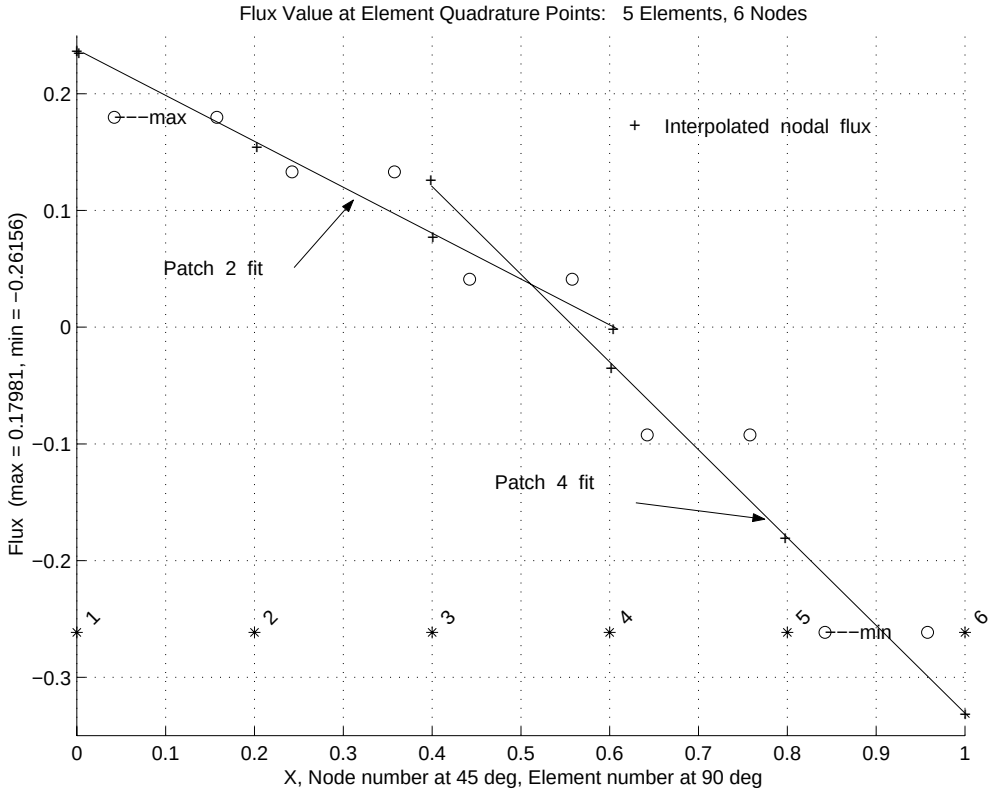


Figure 2.12.6 Element flux and linear patch fits (even elements)

 © 2004 J.E. Akin. All rights reserved.

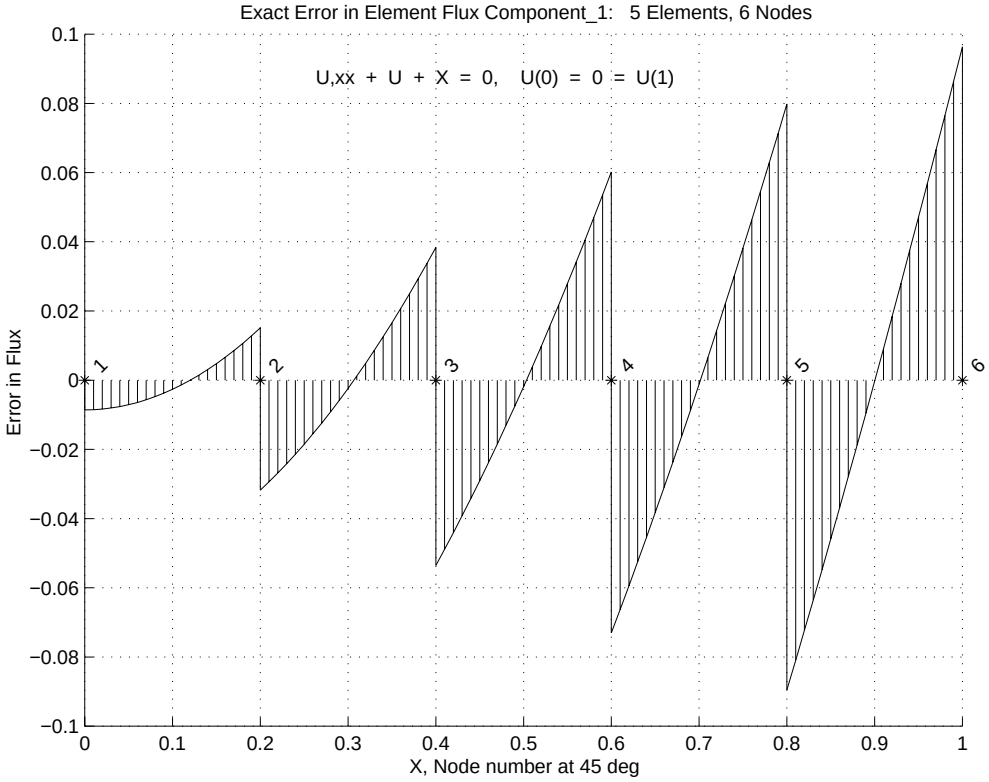


Figure 2.12.7 Exact error in element flux distribution

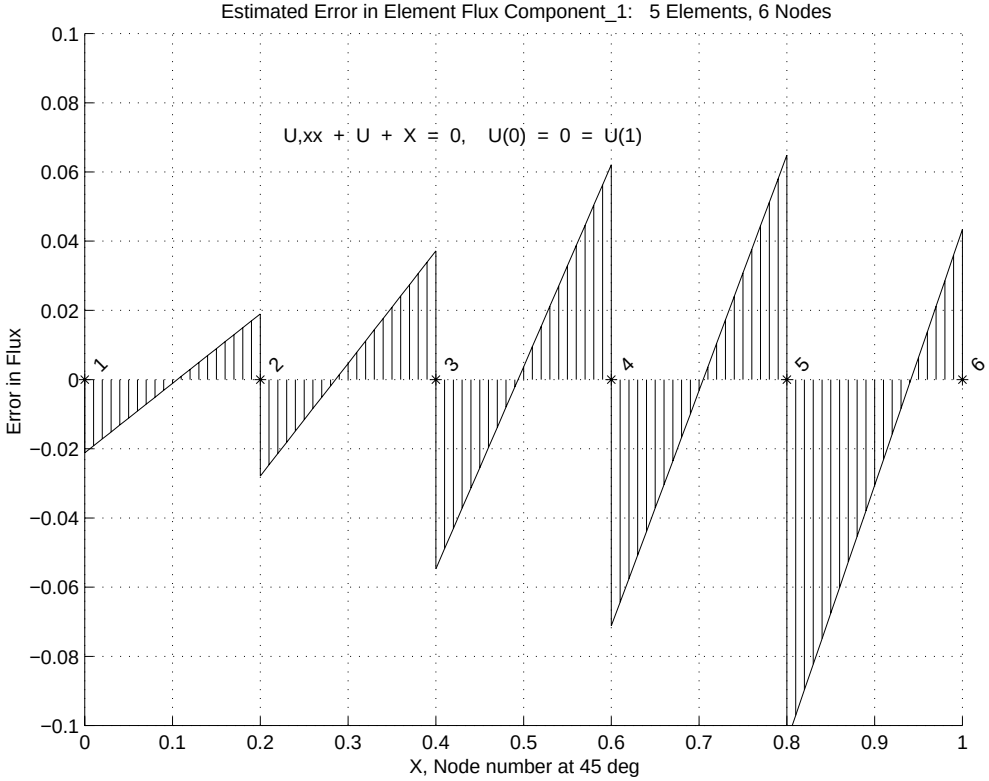


Figure 2.12.8 Estimated error in element flux distribution

 © 2004 J.E. Akin. All rights reserved.

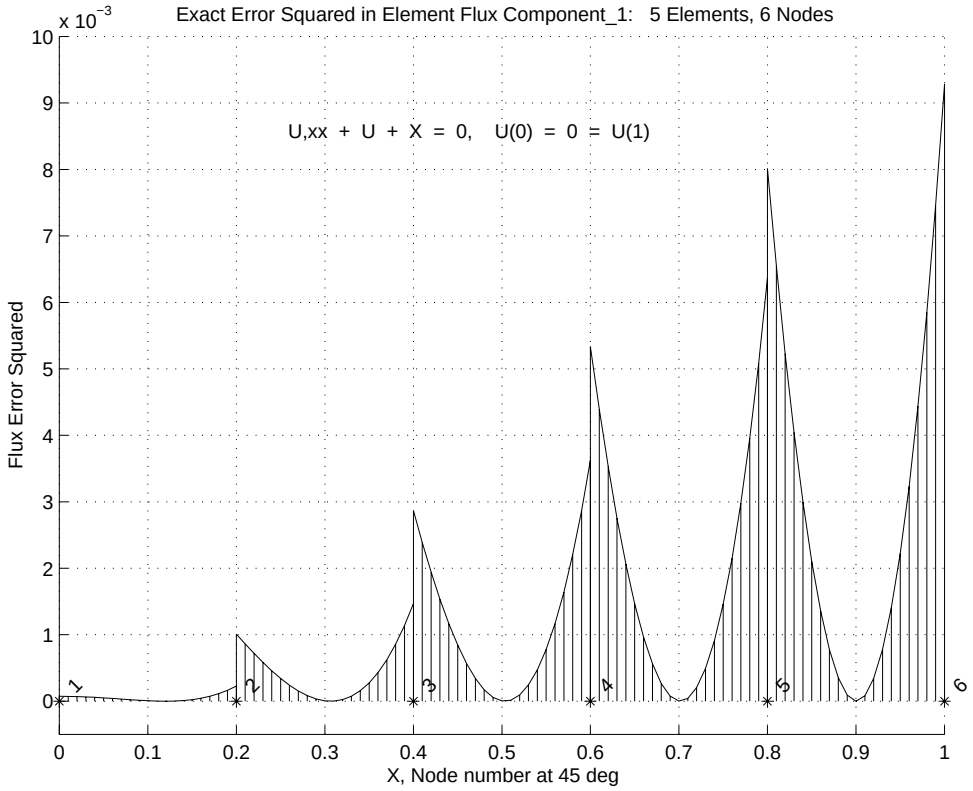


Figure 2.12.9 Square of exact error in element flux distribution

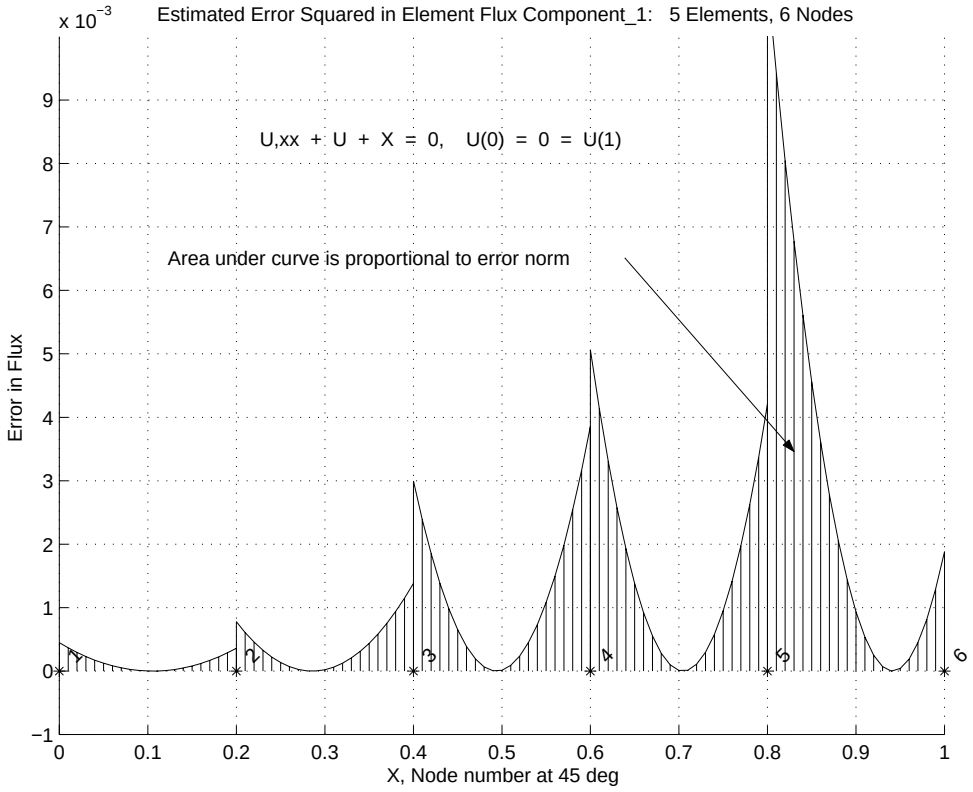


Figure 2.12.10 Square of estimated error in element flux distribution

 © 2004 J.E. Akin. All rights reserved.

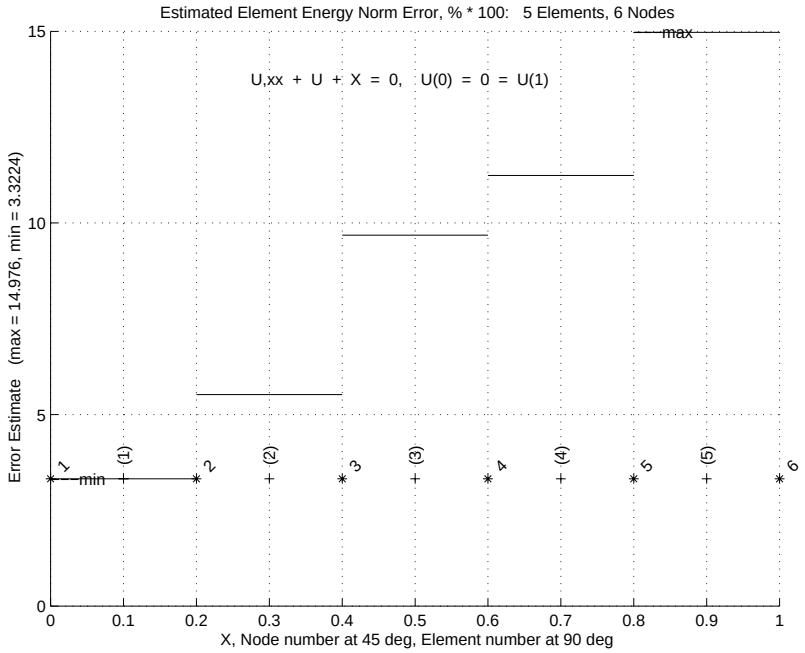


Figure 2.12.11 Estimated energy norm error in each element

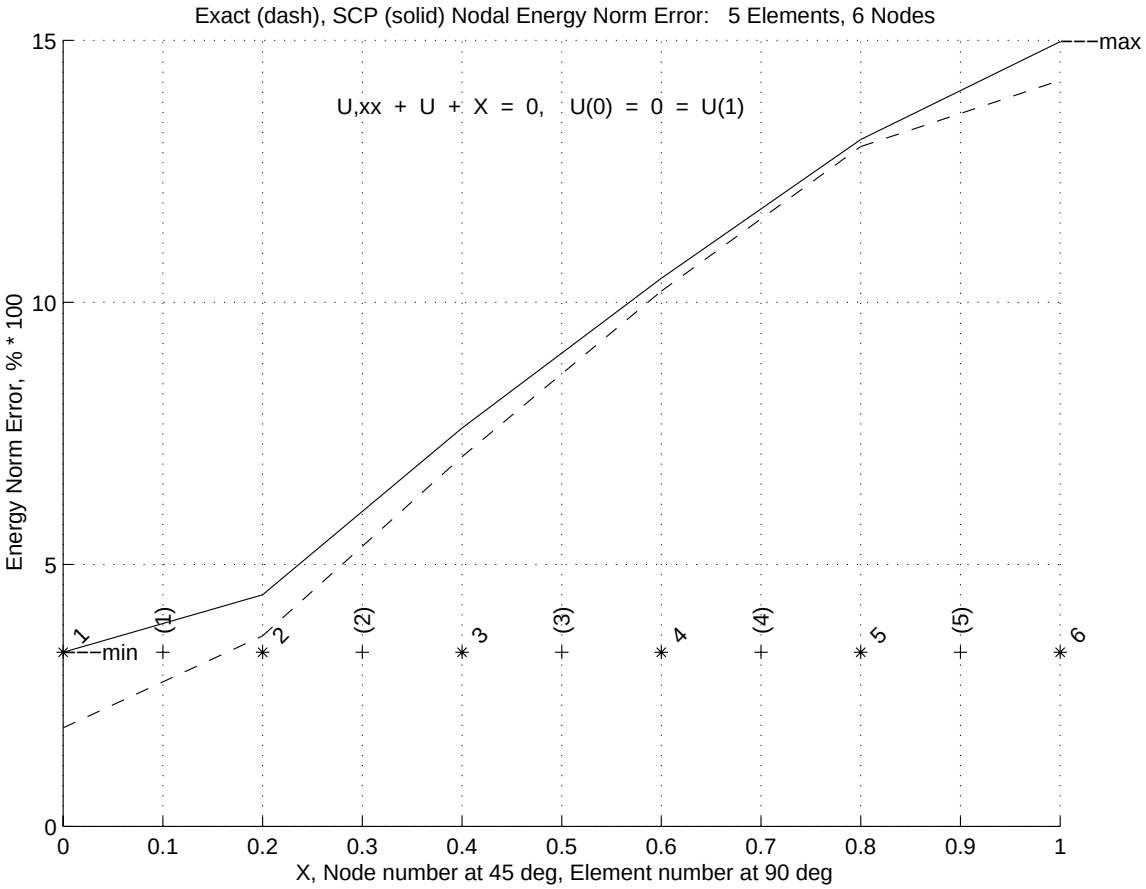


Figure 2.12.12 Exact and averaged energy norm error at the mesh nodes

 © 2004 J.E. Akin. All rights reserved.

distribution. In Fig. 2.11.3, the vertical lines show the exact error in the flux, e_σ. We will see later that it is the quantity whose norm can be used to establish an error indicator. Of course, in practice one does not know the exact flux distribution. However, even for this very crude mesh we can see that the patch averaged flux, $\boldsymbol{\sigma}^*$, is is a much better estimate of the true flux,σ, than are the standard element estimate, $\hat{\boldsymbol{\sigma}} = \mathbf{B}^e\boldsymbol{\phi}^e$. This is confirmed by comparing Fig. 2.11.4 to 2.11.3 where the vertical lines are the difference between the patch averaged and standard element flux values, i.e. $\boldsymbol{\sigma}^* - \hat{\boldsymbol{\sigma}}$.

The flux at each Gauss point is again plotted (as open circles) in Figs. 2.12.5 and 6. Also shown there are solid lines that represent the linear fit (same degree as the assumed element solution) over the elements in each patch. There were five different element patches corresponding to the five elements in the mesh. The first and last patches contained only two elements because the originating elements occurred on the boundary. The interior patches consisted of three elements each: the original element and the adjacent "facing" element on the left and right. Once a patch fit has been obtained it is used to interpolate to the nodal values on that line (marked with a plus symbol). In that process each node in the original mesh receives multiple estimated flux values. Averaging all the estimates from each patch containing the node gives the (solid line) values shown earlier in Fig. 2.12.2. We will assume that the piecewise linear averaged fit for the flux in Fig. 2.12.2 is more accurate that the piecewise constant steps from the original element estimates.

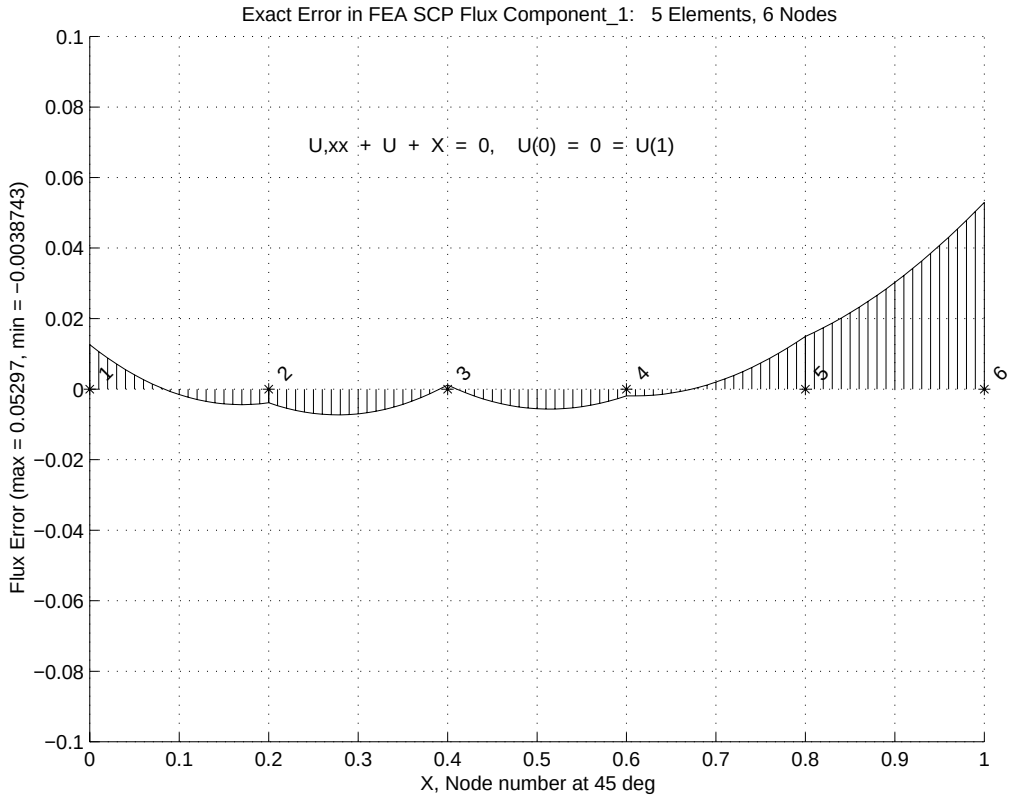


Figure 2.12.13 Exact error in the averaged patch flux estimate

 © 2004 J.E. Akin. All rights reserved.

The exact error in the flux, $\mathbf{e}_\sigma(\mathbf{x}) \equiv \boldsymbol{\sigma}(\mathbf{x}) - \hat{\boldsymbol{\sigma}}(\mathbf{x})$, is shown in Fig. 2.12.7. The estimated error in the flux, $\mathbf{e}_\sigma \approx \boldsymbol{\sigma}^*(\mathbf{x}) - \hat{\boldsymbol{\sigma}}(\mathbf{x})$, obtained from the SCP average flux data is shown to the same scale in Fig. 2.12.8. The comparisons are reasonably good and would improve as the mesh is refined or adapted. Both plots clearly show that the flux error from a standard element calculation,$\hat{\boldsymbol{\sigma}}$, is usually largest at the nodes on the element interfaces. Note that the integral of the square of the above stress error distribution would define a stress error norm. Likewise, each element contributes to that norm and we could compare the local error to the average error to select elements for refinement or de-refinement. If we use the $\mathbf{E}^e$ matrix to scale the product of the stress error we would obtain the more common strain energy norm of the error. In either case the norm can be viewed as being directly proportional to the area under the curve defined by the stress error. Those exact and estimated error norms are shown as the hatched portions of Figs. 2.12.9 and 10, respectively.

Referring to Fig. 2.12.10, the area under the curve for each element can be used to assign a (constant) error value to each element. They are shown in Fig. 2.12.11. After all elements have been assigned an error value those values can be gathered at the nodes of the mesh to give a spatial approximation of the true energy norm error. Figure 2.12.12 illustrates such an averaging process and compares it to a similar average of the exact error. Of course, the exact error in the energy norm is a continuous function and we expect the nodally averaged values would approach the continuous values as the mesh is refined. For the crude mesh used in this example one can actually see the differences in exact and approximate plots, but for a fine mesh they usually look the same an one must rely on the numerical process to obtain useful error estimates.

Above we noted that the standard element level flux estimate, $\hat{\boldsymbol{\sigma}} = \mathbf{E}^e\mathbf{B}^e\boldsymbol{\phi}^e$ is discontinuous at element interfaces and least accurate at those locations. Figure 2.12.13 shows the exact error in the SCP averaged flux estimates for this problem, to the same scale used to give the standard flux error in Fig. 2.12.7. There we see a number of improvements. The flux is continuous at the nodes. Its error is usually smallest at the element interfaces, except for nodes on the domain boundary. Usually the boundary has a significant effect and it is desirable to use smaller elements near the boundary. Special patch processes can be added to try to improve the flux estimates near boundaries but we will not consider such processes. Having illustrated the process in one-dimension we next consider a common two-dimensional test case for error estimators.

2.13 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

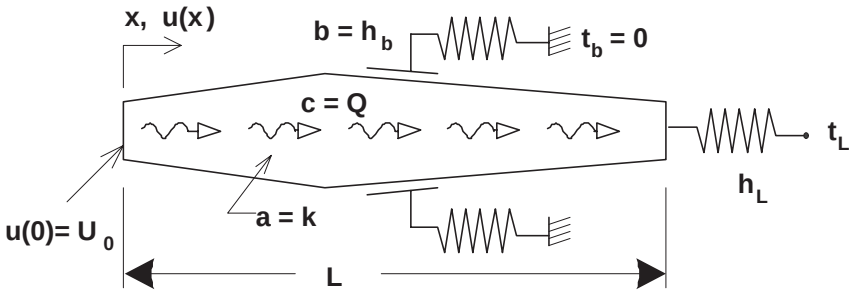
 © 2004 J.E. Akin. All rights reserved.

(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



$$-(a\,u')' + b\,u + c = 0$$

$$\int_L (a\,u'\,v' + b\,u\,v)\,dx = \int_L c\,v\,dx + k_L\,(d_L - u(L))\,v(L)$$

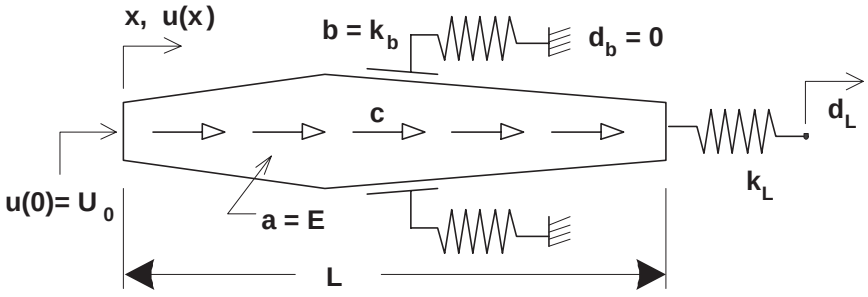


Figure 2.13.1 Rods with a mixed (Robin) boundary condition, from h_L and k_L

 © 2004 J.E. Akin. All rights reserved.

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 condition notation here is: $a\, \partial u(x_R)/\partial n + r_1\, u(x_R) + r_2 = 0$. The most common example of this type of boundary condition is heat convection from a surface where the normal flux is $-k_n(x_R)\, \partial u(x_R)/\partial n - h(x_R)\, [u(x_R) - u_\infty(x_R)] = 0$ so that $r_1 = h$ and $r_2 = -hu_\infty$, where $u(x_R)$ is the unknown temperature on the convecting surface, $u_\infty(x_R)$ is the known surrounding fluid temperature, h is the convection coefficient, and k_n the thermal conductivity normal to the boundary.

In terms of the effects on the corresponding algebraic equation system a Dirichlet condition reduces the number of primary unknowns to be solved. That is, it reduces the size of the effective algebraic system that must be solved (see the following section). However, it also introduces an unknown nodal reaction term in the RHS that can optionally be recovered, as noted below. A Neumann condition does not change the size of the algebraic system; it simply adds known additional terms to the RHS. If that value is zero then no action is required and it is then usually called a natural boundary condition. A Robin, or mixed, condition likewise adds known additional terms to the RHS but, more importantly, it also adds a known symmetric square matrix contribution to the LHS in the rows and columns that correspond to the Robin surface nodes (usually heat convection nodes). For the one-dimensional problems considered in this chapter the Robin, or mixed condition can only occur at an end point (a single dof). For that node number the value of r_1 is added to the diagonal of the system square matrix, and the value of $-r_2$ is added to the corresponding row on the RHS (or for the common convection case add $h^b\, A^b$ and $h^b\, A^b\, u^b_\infty$, respectively for a convecting area A at the point). The corresponding Robin boundary contributions in 2-D will be covered in Section 4.3. Figure 2.13.1 illustrates a thermal (top) and stress problem, in 1-D, with a Dirichlet condition on the left ($x = 0$) end and a Robin condition on the right end ($x = L$).

2.14 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

$$\begin{bmatrix} \mathbf{S}_{uu} & \mathbf{S}_{uk} \\ \mathbf{S}_{ku} & \mathbf{S}_{kk} \end{bmatrix} \begin{Bmatrix} \mathbf{D}_u \\ \mathbf{D}_k \end{Bmatrix} = \begin{Bmatrix} \mathbf{C}_u \\ \mathbf{C}_k + \mathbf{P}_k \end{Bmatrix} \tag{2.48}$$

 © 2004 J.E. Akin. All rights reserved.

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

$$\mathbf{S}_{uu}\,\mathbf{D}_u + \mathbf{S}_{uk}\,\mathbf{D}_k \;=\; \mathbf{C}_u$$

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

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,

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

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

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

In nonlinear and time dependent applications the reactions can be found from similar calculations. 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.

2.15 Elliptic Boundary Value Problems

2.15.1 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 ($n_s = 1$) model problem which will serve as an analytical example:

$$k\,\frac{d^2\phi}{dx^2} \;+\; Q \;=\; 0 \tag{2.49}$$

on the closed domain, $x \in \,]\,0, L\,[\,$, and is subjected to the boundary conditions

 © 2004 J.E. Akin. All rights reserved.

$\phi(L) = \phi_L$, and $k\, d\phi/dx\,(0) = q_0$. 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, ϕ , which satisfies the essential condition, $\phi(L) = \phi_L$, and satisfies

$$I = \int_0^L \left[k\,(d\phi/dx)^2 - \phi\, Q \right] dx + q_0\,\phi(0) - q_L\,\phi(L) = 0, \tag{2.50}$$

also satisfies the new ODE. A typical element contribution is:

$$I^e = \mathbf{D}^{e^T}\,\mathbf{S}^e\,\mathbf{D}^e - \mathbf{D}^{e^T}\,\mathbf{C}^e, \tag{2.51}$$

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 $\mathbf{S}\,\mathbf{D} = \mathbf{C}$ where

$$\mathbf{S} = \sum_{e=1}^{n_e} \boldsymbol{\beta}^{e^T}\,\mathbf{S}^e\,\boldsymbol{\beta}^e, \qquad \mathbf{C} = \sum_{e=1}^{n_e} \boldsymbol{\beta}^{e^T}\,\mathbf{C}^e + \sum_{b=1}^{n_b} \boldsymbol{\beta}^{b^T}\,\mathbf{C}^b,$$

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:

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

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

$$k\,\phi(x) = k\,\phi_L + q(x - L) + Q_0\,(L^2 - x^2)/2. \tag{2.52}$$

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, $\mathbf{S}\,\mathbf{D} = \mathbf{C}$:

$$\frac{2k}{L}\begin{bmatrix} 1 & -1 & 0 \\ -1 & (1+1) & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{Bmatrix} = \frac{Q_0\, L}{4} \begin{Bmatrix} 1 \\ (1+1) \\ 1 \end{Bmatrix} - \begin{Bmatrix} q_0 \\ 0 \\ -q_L \end{Bmatrix}.$$

However, these equations do not yet satisfy the essential boundary condition of $\phi(L) = \phi_3 = \phi_L$. That is, $\mathbf{S}$ is singular and can not be inverted. After applying this essential boundary condition, the reduced equations are

$$\frac{2k}{L}\begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix} \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix} = \frac{Q_2\, L}{4} \begin{Bmatrix} 1 \\ 2 \end{Bmatrix} - \begin{Bmatrix} q \\ 0 \end{Bmatrix} + \frac{2kt_L}{L} \begin{Bmatrix} 0 \\ 1 \end{Bmatrix},$$

or $\mathbf{S}_r\,\mathbf{D}_r = \mathbf{C}_r$. Inverting $\mathbf{S}_r$ and solving for $\mathbf{D}_r = \mathbf{S}_r^{-1}\,\mathbf{C}_r$ yields

 © 2004 J.E. Akin. All rights reserved.

$$\mathbf{S}_r^{-1} = \frac{L}{2k}\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{D}_r = \begin{Bmatrix} \phi_1 \\ \phi_2 \end{Bmatrix} = \frac{Q_0 L^2}{8k}\begin{Bmatrix} 4 \\ 3 \end{Bmatrix} - \frac{qL}{2k}\begin{Bmatrix} 2 \\ 1 \end{Bmatrix} + t_L \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

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. 2.15.1, 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 order $0(L^{e2})$, while all other points are only order $0(L^e)$ 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 $x = L$:

$$\frac{2k}{L}\left[\, 0 - 1\,\phi_2 + 1\,\phi_3 \,\right] = Q_0 \frac{L}{4} - (-q_L)$$

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

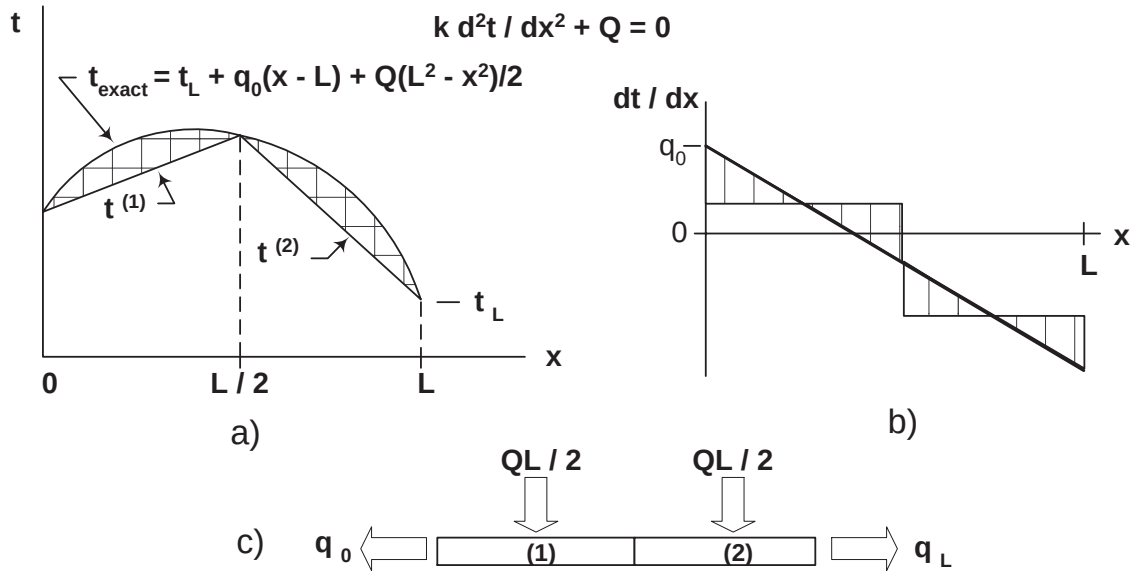


Figure 2.15.1 Example interpolate solution

 © 2004 J.E. Akin. All rights reserved.

one could use programs like *SYSTEM_ROW_SAVE* and *GET_REACTIONS* to store and later recover the associated rows of the square matrix. It is not necessary to save these data when they are first generated since they can clearly be recomputed in a post-processing segment if desired. In the later examples we will invoke functions to save these data as they are created.

2.15.2 Two-Dimensional Equations

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

$$L(u) \;=\; \frac{\partial}{\partial x}\left(k_x \,\frac{\partial u}{\partial x}\right) + \;\frac{\partial}{\partial y}\left(k_y \,\frac{\partial u}{\partial y}\right) - Q - \zeta\,\frac{\partial u}{\partial t} \;=\; 0\,. \tag{2.53}$$

We get the weak form by multiplying by a test function, $w(x)$, and setting the integral to zero. The last two terms are simply

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

Our main interest is with the first two terms

$$I_k \;=\; \int\limits_\Omega \left[\; w\,\frac{\partial}{\partial x}\left(\; k_x \;\frac{\partial u}{\partial x}\right) + \;w\,\frac{\partial}{\partial y}\left(\; k_y \;\frac{\partial u}{\partial y}\right)\right] d\;\Omega$$

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

$$\int\limits_\Omega \left[\;\frac{\partial \Psi}{\partial x} \;-\; \frac{\partial \gamma}{\partial y}\right] d\,\Omega \;=\; \int\limits_\Gamma \Big[\;\gamma\; dx + \Psi\; dy\;\Big]$$

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

$$I_k \;=\; \int\limits_\Omega \left[\;\frac{\partial}{\partial x}\left(\; w\,k_x \;\frac{\partial u}{\partial x}\right) + \;\frac{\partial}{\partial y}\left(\; w\,k_y \;\frac{\partial u}{\partial y}\right)\right] d\;\Omega$$

$$-\int\limits_\Omega \left[\;\frac{\partial w}{\partial x}\;k_x\;\frac{\partial u}{\partial x} \;+\; \frac{\partial w}{\partial y}\;k_y\;\frac{\partial u}{\partial y}\right] d\,\Omega\,.$$

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

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

$$-\int\limits_\Omega \left[\;k_x\,\frac{\partial w}{\partial x}\;\frac{\partial u}{\partial x} \;+\; k_y\,\frac{\partial w}{\partial y}\;\frac{\partial u}{\partial y}\right] d\,\Omega\,.$$

For a contour integral with a unit normal vector, **n**, with components $[\; n_x \quad n_y\;]$, we note the geometric relationships are that (see Fig. 2.15.2) $-dx \;=\; ds\,\mathrm{Cos}\;\theta_y \;=\; ds\,n_y\,,$ and that

 © 2004 J.E. Akin. All rights reserved.

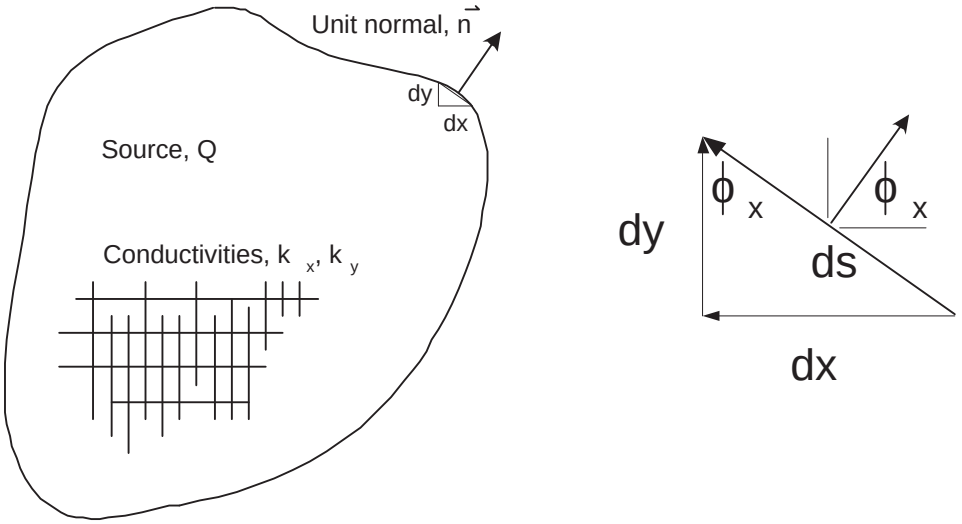


Figure 2.15.2 The unit normal vector components

$dy = ds\,\text{Cos}\;\theta_x = ds\,n_x$ which reduces the boundary integral to

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

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

$$\begin{aligned} -I &= \int_\Omega\left[k_x\frac{\partial w}{\partial x}\frac{\partial u}{\partial x} + k_y\frac{\partial w}{\partial y}\frac{\partial u}{\partial y}\right]d\,\Omega + \int_\Omega Q\,w\,d\,\Omega \\ &+ \int_\Omega \zeta\,w\frac{\partial u}{\partial t}\,d\,\Omega - \int_\Gamma w\,k_n\frac{\partial u}{\partial n}\,ds = 0\,. \end{aligned} \tag{2.54}$$

Note that Green's theorem brought in the information on the conditions of the surface. If we split this into the union of all element domains so that $\Omega = \bigcup_e \Omega^e$ and likewise the boundary becomes the union of the boundary segments created by the mesh, $\Gamma = \bigcup_b \Gamma^b$, we generate four typical matrices from these terms. We interpolate, as before, with the weights $w(x,\,y) = \mathbf{H}^e(x,\,y)\,\mathbf{V}^e$, but include transient time effects by assuming that the degrees of freedom become time dependent, $\mathbf{U}^e = \mathbf{U}^e(t)$, so that $u\,(x,y,t) = \mathbf{H}^e(x,y)\,\mathbf{U}^e(t)$. This assumption for the time behavior makes

$$\frac{\partial u}{\partial t} = \mathbf{H}^e(x,y)\,\frac{\partial}{\partial t}\,\boldsymbol{\phi}^e(t) = \mathbf{H}^e(x,y)\,\dot{\boldsymbol{\phi}}^e\,. \tag{2.55}$$

 © 2004 J.E. Akin. All rights reserved.

```
! ..............................................................  !  1
!    ***  ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS FOLLOW ***  !  2
!      For required REAL (DP) :: S (LT_FREE, LT_FREE)             !  3
!      and optional REAL (DP) :: C (LT_FREE)                      !  4
!      using previous solution D (LT_FREE) with                   !  5
!   D = D_OLD + RELAXATION (D_NEW - D_OLD) via key relaxation 1.0 !  6
!   Globals TIME_METHOD, TIME_STEP, TIME along with mass matrix   !  7
!   options EL_M ((LT_FREE, LT_FREE), and DIAG_M  (LT_FREE).      !  8
! ..............................................................  !  9
!  APPLICATION DEPENDENT Galerkin FOR TRANSIENT PDE via the       ! 10
!  iterative element level assembly of semi-discrete form(s).     ! 11
!  K_e U,xx + Q - Rho_e U,t  = 0, with U(x,0) given by keyword    ! 12
!  start_value, U(0,t) and U(L,t) given by EBC constant in time   ! 13
!  Keywords time_step, time_method, scalar_source, initial_value, ! 14
!  diagonal_mass, etc. are also available.                        ! 15
                                                                  ! 16
REAL(DP)  :: DL, DX_DR, Q = 0.d0 ! Length, Jacobian, source       ! 17
INTEGER   :: IQ, J               ! Loops                          ! 18
                                                                  ! 19
!   Work items for time integration                               ! 20
REAL(DP), SAVE :: Del_t = 1, K_e = 1, Rho_e = 1  ! defaults       ! 21
REAL(DP)  :: K (LT_FREE, LT_FREE), M (LT_FREE, LT_FREE)           ! 22
REAL(DP)  :: F (LT_FREE), WORK (LT_FREE, LT_FREE)                 ! 23
                                                                  ! 24
 IF ( THIS_EL == 1 ) THEN  ! first action for each iteration      ! 25
   IF ( TIME_STEP /= 1.d0 ) Del_t = TIME_STEP  ! non-default      ! 26
   TIME = START_TIME + THIS_ITER * TIME_STEP   ! via globals      ! 27
   PRINT *, '(VIA ITERATION) TIME = ', TIME                       ! 28
 END IF                                                           ! 29
                                                                  ! 30
 K = 0 ;  M = 0 ; F = 0    ! Initialize conduction, mass, source  ! 31
 IF ( EL_REAL > 1 ) THEN   ! use non-default element properties   ! 32
   K_e   = GET_REAL_LP (1) ! thermal conductivity                 ! 33
   Rho_e = GET_REAL_LP (2) ! rho*c_p                              ! 34
 END IF                                                           ! 35
 E     = K_e                                    ! Diffusion       ! 36
 DL    = COORD (LT_N, 1) - COORD (1, 1)         ! Length          ! 37
 DX_DR = DL / 2.                                ! Jacobian        ! 38
 IF ( SCALAR_SOURCE /= 0.d0 ) Q = SCALAR_SOURCE   ! Source        ! 39
                                                                  ! 40
 CALL STORE_FLUX_POINT_COUNT ! Save LT_QP for post-processing     ! 41
 DO IQ = 1, LT_QP            ! LOOP OVER QUADRATURES              ! 42
                                                                  ! 43
!        GET INTERPOLATION FUNCTIONS, AND X-COORD                 ! 44
   H    = GET_H_AT_QP (IQ)   ! SHAPE FUNCTIONS HERE               ! 45
   XYZ  = MATMUL (H, COORD)  ! ISOPARAMETRIC                      ! 46
                                                                  ! 47
!        LOCAL AND GLOBAL DERIVATIVES, B = DGH                    ! 48
   DLH   = GET_DLH_AT_QP (IQ)         ! dH / dr                  ! 49
   DGH   = DLH / DX_DR                ! dH / dx                  ! 50
                                                                  ! 51
!       SQUARE MATRICES (STIFFNESS & MASS) & SOURCE VECTOR       ! 52
   K = K + MATMUL (TRANSPOSE(DGH), DGH)* WT (IQ) * DX_DR * K_e    ! 53
   M = M + OUTER_PRODUCT (H, H)        * WT (IQ) * DX_DR * Rho_e  ! 54
   F = F + H * Q                       * WT (IQ) * DX_DR          ! 55
                                                                  ! 56
   CALL STORE_FLUX_POINT_DATA (XYZ, E, DGH) ! for post-processing ! 57
 END DO  ! QUADRATURE                                             ! 58
```

Figure 2.16.1a Element matrices before time step changes

 © 2004 J.E. Akin. All rights reserved.

```
                                                                          ! 59
!  Assemble for semi-discrete one-step time rule, via iteration:          ! 60
                                                                          ! 61
!  NOTE: This is very inefficient since the assembly, boundary            ! 62
!  conditions, and factorization are repeated ever "time step".           ! 63
                                                                          ! 64
 IF ( DIAGONAL_MASS ) THEN  ! use the scaled diagonal mass form           ! 65
   CALL DIAGONALIZE_SQ_MATRIX (LT_FREE, M, DIAG_M) ; M = 0.d0             ! 66
   DO J = 1, LT_FREE                                                     ! 67
     M (J, J) = DIAG_M (J)                                               ! 68
   END DO                                                                ! 69
 END IF ! diagonal instead of consistent or averaged mass matrix         ! 70
                                                                          ! 71
 SELECT CASE ( TIME_METHOD ) ! for one-step time integration rule        ! 72
   CASE ( 2 ) ! Crank-Nicolson, accurcy order O(Del_t^2)                 ! 73
      WORK = M / Del_t - K / 2                                           ! 74
      C    = F + MATMUL (WORK, D)                                        ! 75
      S    = M / Del_t + K / 2                                           ! 76
   CASE ( 3 ) ! Galerkin in time, accurcy order O(Del_t^2)               ! 77
      WORK = M / Del_t - K / 3                                           ! 78
      C    = F + MATMUL (WORK, D)                                        ! 79
      S    = 2 * K / 3.d0 + M / Del_t                                    ! 80
   CASE ( 4 ) ! Least Squares in time, F constant in time                ! 81
      WORK =  MATMUL (TRANSPOSE(M), M) / Del_t     &                     ! 82
           +  MATMUL (TRANSPOSE(K), M) / 2         &                     ! 83
           -  MATMUL (TRANSPOSE(M), K) / 2         &                     ! 84
           -  MATMUL (TRANSPOSE(K), K) * Del_t / 6                       ! 85
      C    = -MATMUL (TRANSPOSE(K), F) * Del_t / 2 &                     ! 86
           -  MATMUL (TRANSPOSE(M), F)             &                     ! 87
           +  MATMUL (WORK, D)                                           ! 88
      S    =  MATMUL (TRANSPOSE(K), K) * Del_t / 3 &                     ! 89
           +  MATMUL (TRANSPOSE(K), M) / 2         &                     ! 90
           +  MATMUL (TRANSPOSE(M), K) / 2         &                     ! 91
           +  MATMUL (TRANSPOSE(M), M) / Del_t                           ! 92
   ! Add generalized trapezoidal method as CASE ( 5 )                    ! 93
   CASE DEFAULT ! Method 1, forward difference, order O(Del_t)           ! 94
      WORK = M / Del_t                                                   ! 95
      C    = F + MATMUL (WORK, D)                                        ! 96
      S    = K + M / Del_t                                               ! 97
 END SELECT ! a one-step rule                                            ! 98
                                                                          ! 99
!    ***  END ELEM_SQ_MATRIX PROBLEM DEPENDENT STATEMENTS ***            !100
```

Figure 2.16.1ab New element matrices after time step changes

These assumptions result in two square matrices

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

and the two source vectors

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

which when assembled yield a system of ordinary differential equations in time such that $\mathbf{V}^T \, (\mathbf{M} \, \dot{\boldsymbol{\phi}} + \mathbf{K} \, \boldsymbol{\phi} - \mathbf{F}) = 0$, so that for arbitrary $\mathbf{V} \neq \mathbf{0}$, we have

$$\mathbf{M} \, \dot{\boldsymbol{\phi}}(t) + \mathbf{K} \, \boldsymbol{\phi}(t) = \mathbf{F}(t), \tag{2.56}$$

 © 2004 J.E. Akin. All rights reserved.

which is an initial value problem to be solved from the initial condition state at $\boldsymbol{\phi}(x, y, t = 0)$. Here we will note that $\mathbf{M}^e$ and $\mathbf{M}$ are usually called the *element* and *system mass* (or *capacity*) *matrices*, respectively. By making assumptions about how to approximate the time dependent vector $\dot{\boldsymbol{\phi}}(t)$ this system can be rearranged to yield a linear system $\mathbf{S}\,\boldsymbol{\phi}(t) = \mathbf{C}(t)$ to be solved at each time step for the current value of $\boldsymbol{\phi}(t)$. For the steady state case, $\partial / \partial t = 0$, this reduces to the system of algebraic equations $\mathbf{S}\,\boldsymbol{\phi} = \mathbf{C}$ considered earlier. The same procedure is easily carried out in three dimensions. Matrix $\mathbf{K}^e$ changes by having a third term involving z and for $\mathbf{F}_q^b$ we think of $d\Gamma$ as a surface area instead of a line segment.

We also note in passing that in addition to using classical finite differences in time there are two other choices for methods to treat time dependence. One is to do a separation of variables and also interpolate the element behavior over one or more time steps with $u(x, y, t) = \mathbf{H}^e(x, y)\,\mathbf{h}^e(t)\,\boldsymbol{\phi}^e$. [30] That allows one to carry out weighted residual methods in time also. These first two choices are usually referred to as "semi-discrete" methods for time integration. Another approach is to carry out a full space-time interpolation with $u(x, y, t) = \mathbf{H}^e(x, y, t)\,\boldsymbol{\phi}^e$ [6, 7] and include three-dimensional space via four-dimensional elements [26]. The 4-D element solutions have very large memory requirements (by todays standards), and are usually restricted to a time slab (a constant time step) which simplifies the 4-D elements and the associated 4-D mesh generation. Like the above example, most 4-D models begin with the classical mechanics laws in 3-D space and then interpolate through time also. It is also possible to begin the formulation using relativity physics laws in a 4-D space-time and to form the 4-D elements therein [16]. Such an approach is not required unless the velocities involved are not small relative to the speed of light.

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.*

2.16 Initial Value Problems

Many applications, like Eq. 2.53, involve the first partial derivative with respect to time and get converted through the finite element process to a matrix system of ordinary differential equations in time, like Eq. 2.56. These are known as initial value problems, or parabolic problems, and require the additional description of the spatial value of $\boldsymbol{\phi}$ at the starting time (known as the initial condition). There are hundreds of ways to numerically integrate these matrices in time. One has to be concerned about the relative costs (storage and operation counts), stability, and accuracy of the chosen method. These topics are covered in many books on numerical methods and we will not go into them here. Instead we will simply give some insight to a few approaches.

If one assumes that Eq. 2.56 is valid at an element level for time t_s one could write

 © 2004 J.E. Akin. All rights reserved.

$$\mathbf{M}^e\,\dot{\boldsymbol{\phi}}^e(t_s) + \mathbf{K}\,\boldsymbol{\phi}^e(t_s) \;=\; \mathbf{F}^e\,(\,t_s\,)\,,$$

and assume that at step number s

$$\dot{\boldsymbol{\phi}}_s = \frac{\boldsymbol{\phi}_{s+1} - \boldsymbol{\phi}_s}{\Delta t_{s+1}}$$

where Δt is the time step size, and contemplate rewriting the element relations as

$$[\mathbf{M}^e \,/\, \Delta t_{s+1}]\boldsymbol{\phi}^e_{s+1} \;=\; [\mathbf{M}^e \,/\, \Delta t_{s+1} - \mathbf{K}^e]\boldsymbol{\phi}^e_s \;+\; \mathbf{F}^e_s\,,$$

or as $\mathbf{S}^e(\Delta t_{s+1})\,\boldsymbol{\phi}^e_{s+1} \;=\; \mathbf{C}^e_s(\Delta t_{s+1,} t)$. Then the effective element square matrix would not change with time if the step size, Δt_{s+1} were held constant.

One could begin the time stepping with the initial condition, $\boldsymbol{\phi}_0$, and loop through the time steps, in an iterative fashion, to get the next $\boldsymbol{\phi}_{s+1}$. While this type of concept is illustrated in the element level calculations of Fig. 2.16.1 it is very inefficient to use this element looping (or iteration) process. It is more proper to carry out a similar process after the proper system assembly of Eq. 2.56, invoke the system initial conditions, and solve for $\boldsymbol{\phi}_{s+1}$, after applying the essential boundary conditions and/or flux conditions both of which can now have time dependent values. We will omit that level of detail here (see for example [3]) and simply note that control ketword *transient* is available in *MODEL* to carry out a transient study. It defaults to zero initial conditions and the Crank-Nicolson time integration scheme which is unconditionally stable and second order accurate in time, $O(\Delta t^2)$. The application of transient methods will be given in Chapter 11, but the main interest in this book is on spatial error estimates. The book by Huang and Usmani [13], covers two-dimensional transient error estimators in more detail that space allows here.

However, to give a brief overview of how all the transient calculations proceed we will use a simple (and slightly corrected) example from Desai [11] , where he used 3 linear elements to solve the transient heat conduction problem of

$$\alpha\,\frac{\partial^2 \phi}{\partial x^2} = \frac{\partial \phi}{\partial t}\,,$$

where $\alpha \;=\; k \,/\, \rho c_p$ is the thermal diffusivity, ϕ is temperature, t is time, and x is position. The bar has an initial condition of zero temperature, $\phi(x, 0) = 0$, when the two ends suddenly have different non-zero temperatures enforced and held constant with time, $\phi(0, t) = 10,\ \phi(L, t) = 20$. The element size and time step size are chosen to make the element Fourier Number, $Fo \;=\; \alpha \Delta t \,/\, \Delta x^2$, equal to unity. Here $\Delta x \;=\; L^e \;=\; 1$ (so $L = 3$) and the time step is also taken as unity, $\Delta t = 1$. The thermal stiffness and mass matrix for each element are

$$\mathbf{K}^e = \frac{\alpha}{L^e}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{M}^e = \frac{L^e}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},$$

and the element source vector is null, $\mathbf{C}^e_Q = \mathbf{0}$. Applying the Euler (forward difference) one step time integration rule for a uniform mesh gives system equations

$$\mathbf{K}\{\boldsymbol{\Phi}\}_{s+1} \;+\; \frac{1}{\Delta t}\,\mathbf{M}\{\boldsymbol{\Phi}\}_{s+1} \;=\; \{\mathbf{C}_Q\}_{s+1} \;+\; \frac{1}{\Delta t}\,\mathbf{M}\{\boldsymbol{\Phi}\}_s \;+\; \{\mathbf{C}_q\}_{s+1}\,.$$

Selecting the above numerical values, so $Fo \;=\; 1$, and assemblying the four equations,

 © 2004 J.E. Akin. All rights reserved.

including the initial condition vector, $\mathbf{\Phi}_0$, but before applying the essential boundary conditions at the first and last nodes gives:

$$\frac{1}{6}\begin{bmatrix} 8 & -5 & 0 & 0 \\ -5 & 16 & -5 & 0 \\ 0 & -5 & 16 & -5 \\ 0 & 0 & -5 & 8 \end{bmatrix}\begin{Bmatrix} \Phi_1 = 10 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 = 20 \end{Bmatrix}_1 = \mathbf{C}_Q + \frac{1}{6}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{Bmatrix}_0 + \begin{Bmatrix} -q_1 \\ 0 \\ 0 \\ q_4 \end{Bmatrix}_1$$

where the initial condition nodal vector is zero here, $\mathbf{\Phi}_0 = 0$. Applying the essential boundary conditions at this (and all) time steps (with EBC identities inserted in rows 1 and 4 after saving the originals for later reaction recovery) gives

$$\frac{1}{6}\begin{bmatrix} 6 & 0 & 0 & 0 \\ 0 & 16 & -5 & 0 \\ 0 & -5 & 16 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}\begin{Bmatrix} \Phi_1 = 10 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 = 20 \end{Bmatrix}_1 = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} + \frac{-10}{6}\begin{Bmatrix} -6 \\ -5 \\ 0 \\ 0 \end{Bmatrix} + \frac{-20}{6}\begin{Bmatrix} 0 \\ 0 \\ -5 \\ -6 \end{Bmatrix} = \frac{1}{6}\begin{Bmatrix} 60 \\ 50 \\ 100 \\ 120 \end{Bmatrix}$$

which is solved for the first time step result of

$$\mathbf{\Phi}_1^T = [\,10.00 \quad 5.63 \quad 8.01 \quad 20.00\,].$$

For the second time step we have system equations of

$$\frac{1}{6}\begin{bmatrix} 8 & -5 & 0 & 0 \\ -5 & 16 & -5 & 0 \\ 0 & -5 & 16 & -5 \\ 0 & 0 & -5 & 8 \end{bmatrix}\begin{Bmatrix} \Phi_1 = 10 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 = 20 \end{Bmatrix}_2 = \frac{1}{6}\begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}\begin{Bmatrix} 10.00 \\ 5.63 \\ 8.01 \\ 20.00 \end{Bmatrix} = \frac{1}{6}\begin{Bmatrix} 25.63 \\ 40.53 \\ 57.67 \\ 48.01 \end{Bmatrix}.$$

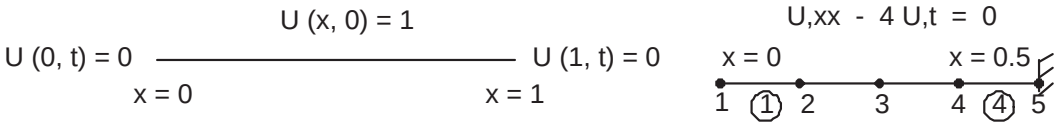
Applying the essential boundary conditions at this time gives the system equations

$$\frac{1}{6}\begin{bmatrix} 6 & 0 & 0 & 0 \\ 0 & 16 & -5 & 0 \\ 0 & -5 & 16 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}\begin{Bmatrix} \Phi_1 = 10 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 = 20 \end{Bmatrix}_2 = \frac{1}{6}\begin{Bmatrix} 60.00 \\ 90.43 \\ 157.67 \\ 120.00 \end{Bmatrix},$$

so $\mathbf{\Phi}_2^T = [\,10.00 \quad 9.68 \quad 12.88 \quad 20.00\,]$ and so on for each later step. Likewise, as done before we can recover the heat flux reactions necessary to maintain the essential boundary conditions at each step. For the first time step they can be shown to be $q_1 = 8.64$, and $q_4 = 19.99\,BTU/sec$ while at the second step they were $q_1 = 0.99$, and $q_4 = 7.93$. These eventually transition to the steady state reactions of $q_1 = -3.33$, and $q_4 = 3.33$ after about 11 steps. The corresponding steady state temperatures are linear between the two essential boundary conditions $\mathbf{\Phi}^T = [\,10.00 \quad 13.33 \quad 16.67 \quad 20.00\,]$.

To illustrate a simple one-dimensional initial value problem consider a uniform bar that is initially at a temperature of unity when suddenly the two ends are reduced to a zero temperature and we want to see the time history of the bar as it cools toward zero everywhere. The analytic solution is know for this widely used example, and is include in the *MODEL* library as *exact_case* 34. Here we use a half-symmetry model with 5 nodes and 4 linear elements. The natural BC occurs at the center point. A set of sample data for this problem are given in Fig. 2.16.2 and the nodal time histories are shown in

 © 2004 J.E. Akin. All rights reserved.




```
title "L2 Solution K_e U,xx - Rho_e U,t = 0, Myers values"       !  1
example      122 ! Application source code library number        !  2
exact_case    34 ! Analytic solution for list_exact              !  3
list_exact       ! List given exact answers at nodes, etc        !  4
transient        ! Problem is first order in time                !  5
save_pt_ans      ! Create node_results.tmp for matlab            !  6
save_exact       ! Save exact result to exact_node_solution.tmp  !  7
save_1248        ! Save after steps 1, 2, 4, 8, 16 ...           !  8
time_groups   1  ! Number of groups of constant time_steps       !  9
time_method   3  ! 1-Euler, 2-Crank-Nicolson, 3-Galerkin, 4-L Sq ! 10
time_steps    64 ! Number of time steps                          ! 11
start_value   1. ! Initial value of transient scalar everywhere  ! 12
time_step  1.d-2 ! Time step size for time dependent solution    ! 13
diagonal_mass    ! Use diagonalized mass matrix                  ! 14
# average_mass   ! Average consistent & diagonal mass matrices   ! 15
bar_chart        ! Include bar chart printing in output          ! 16
no_scp_ave       ! Do NOT get superconvergent patch averages     ! 17
no_error_est     ! Do NOT compute SCP element error estimates    ! 18
nodes       5 ! Number of nodes in the mesh                      ! 19
elems       4 ! Number of elements in the system                 ! 20
dof         1 ! Number of unknowns per node                      ! 21
el_nodes    2 ! Maximum number of nodes per element              ! 22
el_real     2 ! Number of real properties per element            ! 23
el_homo       ! Element properties are homogeneous               ! 24
space       1 ! Solution space dimension                         ! 25
b_rows      1 ! Number of rows in the B (operator) matrix        ! 26
shape       1 ! Element shape, 1=line, 2=tri, 3=quad, 4=hex      ! 27
gauss       2 ! Maximum number of quadrature points              ! 28
remarks     5 ! Number of user remarks                           ! 29
quit ! keyword input, remarks follow                             ! 30
APPLICATION DEPENDENT Galerkin FOR TRANSIENT SOLUTION            ! 31
 K U,xx - R U,t  = 0, with U(x,0)=1, U(0,t)=0, U(L,t)=0          ! 32
Myers/Akin time integration example, Fig 17.2.3 , L_e = 1/8      ! 33
Time integrations: 1-Euler, 2-Crank-Nicolson, 3-Galerkin in time ! 34
K, R default to 1 else real el properties 1 & 2, R= 1/(16 L_eˆ2) ! 35
 1 1 0.            ! node, bc_flag, x                            ! 36
 2 0 0.125         ! node, bc_flag, x                            ! 37
 3 0 0.25          ! node, bc_flag, x                            ! 38
 4 0 0.375         ! node, bc_flag, x                            ! 39
 5 0 0.5   ! natural BC at this center node                      ! 40
    1    1    2    ! elem, two nodes                             ! 41
    2    2    3    ! elem, two nodes                             ! 42
    3    3    4    ! elem, two nodes                             ! 43
    4    4    5    ! elem, two nodes                             ! 44
   1   1 0. ! node, dof, essential value                         ! 45
1 1. 4.0    ! el, K_e, Rho_e                                     ! 46
```

Figure 2.16.2 Data for sudden cooling of a bar

 © 2004 J.E. Akin. All rights reserved.

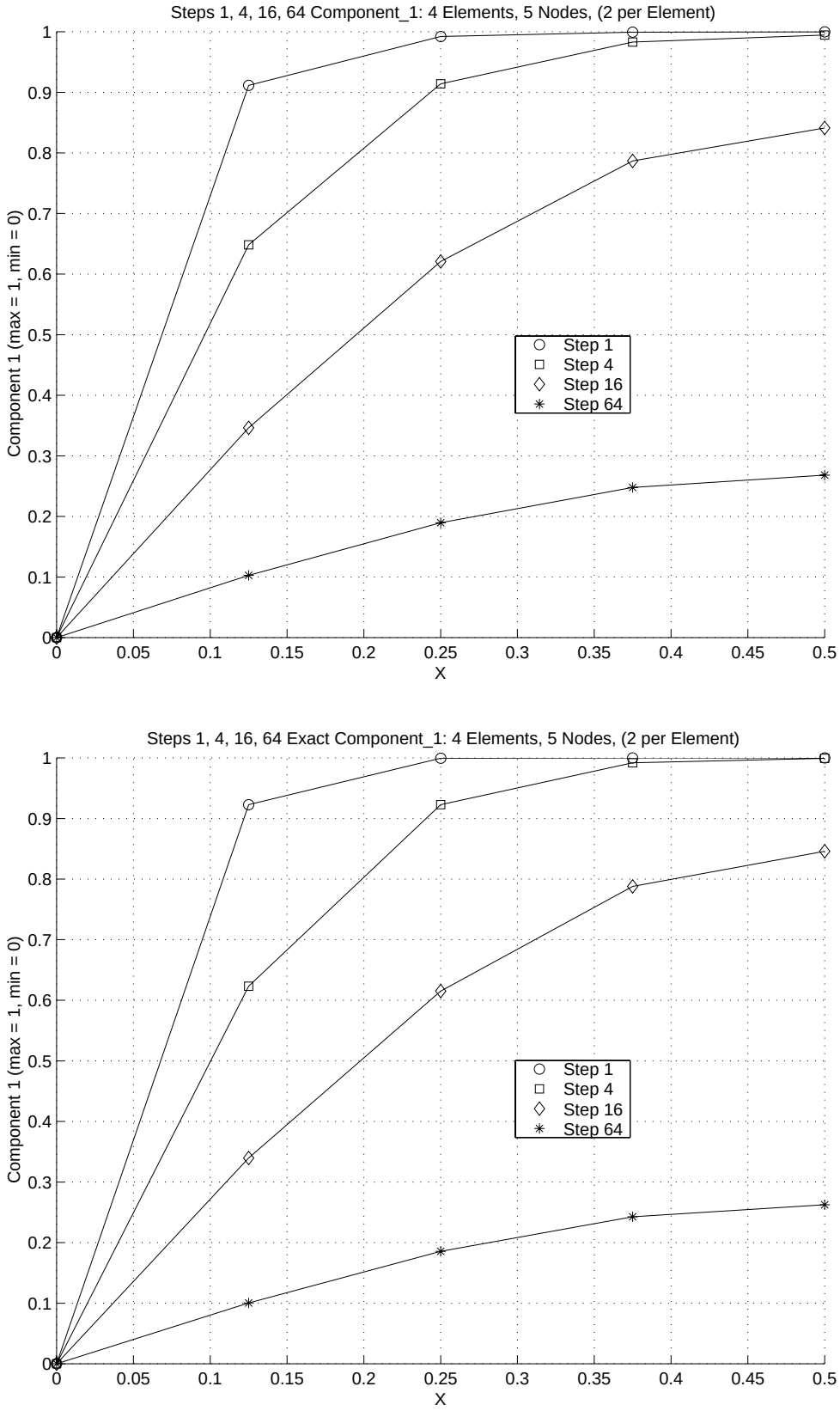


Figure 2.16.3 Cooling bar finite element and exact nodal time-histories

 © 2004 J.E. Akin. All rights reserved.

Fig. 2.16.3 from this crude model and from the exact soultion (evaluated only at the nodes). The early time history of the temperature at the center (symmetry) point is given in Fig. 2.16.4 for a diagonal mass matrix option. The default finite element formulation employes the full consistent mass matrix. In this example we have used a diagonal form (see line 14 of Fig. 2.16.2). The average of the two approaches has been demonstrated to be better for some element families, like the L2 element used here. The example used the numerically integrated element matrices and thus could have used quadratic or cubic elements as well.

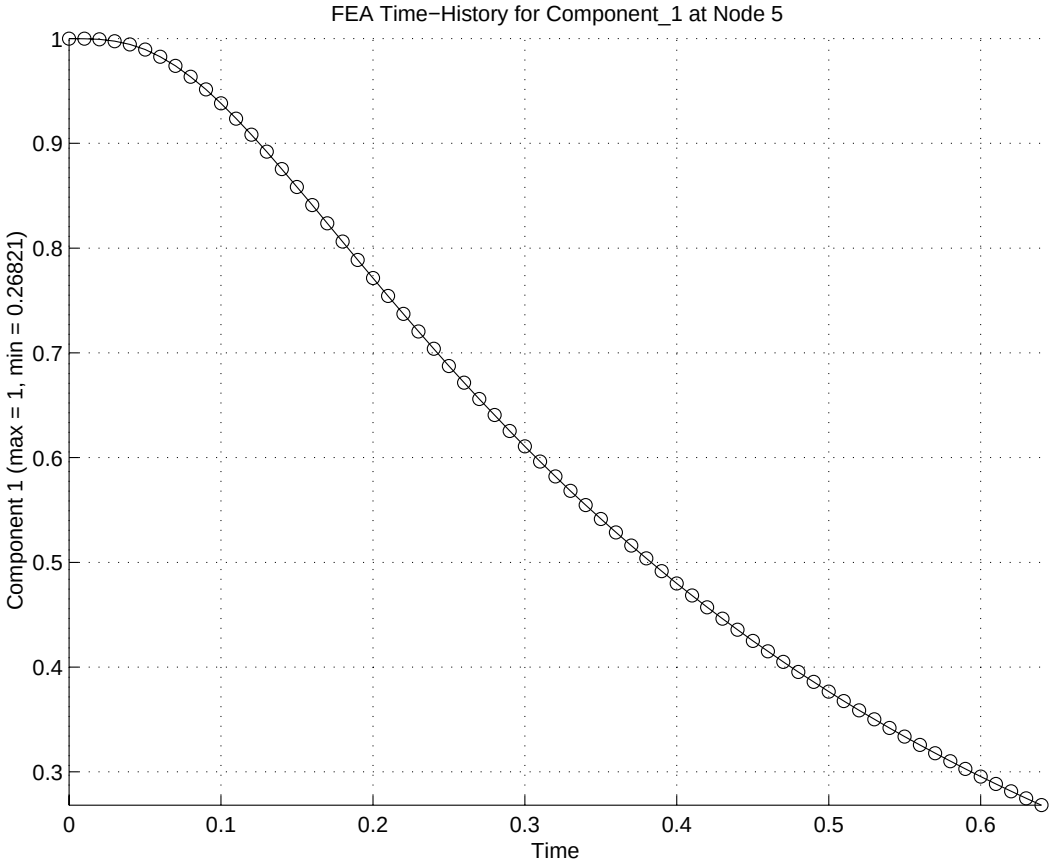


Figure 2.16.4 Cooling bar center point early time history

2.17 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). Consider the one-

 © 2004 J.E. Akin. All rights reserved.

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 $u(0) = 0,\ u(1) = g$. The Hilbert space properties that the function, u, and its first derivative must be square integrable

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

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

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

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

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* $a(u, v)$ and the *linear form* $(\, f, v\,)$:

$$a(u,v) \equiv < u',\ v' > \ = \int_0^1 u'\, v'\, dx, \quad (\, f, v\,) \equiv \int_0^1 f\, v\, dx\,.$$

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

 © 2004 J.E. Akin. All rights reserved.

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

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, $(W) \Longleftrightarrow (S)$ 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, $v \in S$. Now set $\omega = v - u$ such that $v = \omega + u$, and $\omega \in V$. Given $F(v) = \frac{1}{2} < v', v' > - < f, v >$ and expanding

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

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

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 we have both $a(u_1, v) = (f, v)$, and $a(u_2, v) = (f, v)$ for all $v \in V$ and subtracting the second from the first result yields $a(u_1 - u_2, v) = 0$, so $< (u_1' - u_2'), v' > = 0$ for all $v \in V$. Consider the choice $v(x) \equiv u_1(x) - u_2(x)$.

 © 2004 J.E. Akin. All rights reserved.

Is it in V? We note that $v(0) = 0 - 0 = 0$ and $v(1) = g - g = 0$, so it is in V and 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 $u_1(x) - u_2(x) = c$, and 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.

2.18 Exercises

1. The example ordinary differential equation (ODE) $d^2u/dx^2 + u + x = 0$ with $u(0) = 0 = u(1)$ 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 below + 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 $u'' + x^n = 0$, for $x \varepsilon\ ]0, 1[$, with $u(0) = 0 = u(1)$. Assume a cubic polynomial that satisfies the two boundary conditions:

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

Form the **S** and **C** matrices. Solve for **D** from **S D** = **C**. Compare to the exact solution for a) $n = 0$, b) $n = 1$, c) $n = 2$ where $u_{exact} = (x - x^{n+2})/[(n + 1)(n + 2)]$.

3. For the one-dimensional problems, with constant coefficients, we often need to evaluate the following four integrals:

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

 © 2004 J.E. Akin. All rights reserved.

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

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":

$$\boldsymbol{H}(x) = \lfloor H_1(x) \qquad H_2(x) \rfloor$$

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 $q = \partial u/\partial n = \bar{q}(x)$ on Γ_2 where Γ_1 and Γ_2 are non-overlapping parts of the total boundary $\Gamma = \Gamma_1 \bigcup \Gamma_2$. 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:

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

(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

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

(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

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

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. 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 $x \le 0.5$.

7. If the model equation Problem 1 changes to $d[a(x)\,du/dx]/dx + b(x)u + Q(x) = 0$

 © 2004 J.E. Akin. All rights reserved.

show that the $\mathbf{S}^e$,$\mathbf{M}^e$, and $\mathbf{C}^e$ matrices of Problem 3 change to

$$\boldsymbol{S}^e = \int_{L^e} \frac{d\boldsymbol{H}^T}{dx}\, a(x)\, \frac{d\boldsymbol{H}}{dx}\, dx, \quad \boldsymbol{M}^e = \int_{L^e} \boldsymbol{H}^T\, b(x)\, \boldsymbol{H}\, dx, \quad \boldsymbol{C}^e = \int_{L^e} \boldsymbol{H}^T\, Q(x)\, dx.$$

8. If the model equation in Problem 1 changes to $d^2u/dx^2 + \lambda u = 0$ with $u(0) = 0 = u(1)$, where λ is an unknown global constant, how does the $\mathbf{M}^e$ matrix change? How does the classification of the assembled algebraic equations change?

9. For the global approximation examples of Section 2.5 plot, or accurately sketch, the weight function, $w(x)$, for each of the five methods.

10. The first two linear element Galerkin model example yielded the algebraic equation system, of Eq. 2.43, which has 3 equations with 5 unknowns. The system is singular until 2 boundary conditions are supplied to define a unique problem. Employ a pair of Dirichlet and Neumann conditions such that $u(0) = 0$ and $du(1)/dx = 0$ so that the new exact solution is $u(x) = \text{Sin}\,(x) \,/\, \text{Cos}\,(1) - x$. In other words, set $D_1 = 0$ and $q_L = 0$. a) From the algebraic system compute D_2 and D_3 and then the reaction q_0. Compare them to the exact values. b) Post-process both elements to compute the element flux, du/dx. c) Sketch the exact and approximate solution versus position, x. d) Sketch the exact and approximate flux versus position, x.

11. For the two element mesh in Fig. 2.10.3 use Boolean arrays and matrix multiplication and addition to assemble the given example results in Eq. 2.43.

12. Heat conduction through a layered wall is modeled by $k\, d^2u/dx^2 = 0$, where k is the thermal conductivity. With two essential boundary conditions the linear (L2) element will yield exact results, when nodes are placed on any internal material interfaces. A furnace wall has inside and outside temperatures of 1500 F and 150 F, respectively, and it is made of firebrick, insulator, and red brick having conductivities of 0.72, 0.08, and 0.5 BTU/hr ft F, respectively. The corresponding layer thicknesses are 9, 5, and 7.5 inches. Use three unequal length elements to find the internal interface temperatures and the two wall reaction heat fluxes. Post-process the elements for their gradient, and for the flux $q = -k\, du/dx$. Hint, we expect the same q value in each element.

13. Use three equal length elements to solve the problem in Fig. 2.10.6 ($L^e = 1/3$). Obtain the nodal values, reactions, and post process for the element gradients. Compare the nodal values to the exact solution.

14. The transverse deflection, v, of a thin beam is given by the fourth order ODE: $d^2[EI(x)d^2v/dx^2]/dx^2 = p(x)$, where E is the material modulus of elasticity, I is the moment of inertia of the cross-section, and p is a distributed load per unit length. Obtain a Galerkin integral form by integrating the first term twice by parts. In the boundary terms we usually call $\Theta = dv/dx$ the slope, $M = EI\, d^2v/dx^2$ the moment (or couple) at a point, and $F = EI\, d^3v/dx^3$ the transverse shear (or force) at the point.

15. Solve Eq 2.43 for a non-zero Neumann condition of $du/dx(1) = Cotan(1) - 1$ and $u(0) = 0$, and compare the results to the same exact solution.

16. A system governed by $d^2u/dx^2 + Q(x) = 0$ with $u(0) = 0 = u(1)$ has a discontinuous

 © 2004 J.E. Akin. All rights reserved.

source value of $Q = 1$ for $x \le 1/2$ and 0 for $x > 1/2$. a) Explain why it is preferable for an element end to be placed at $x = 1/2$. b) If you used 3 linear (L2) elements explain why it is better to place the two smaller elements in the left half of the domain. c) If you use that mesh verify that the (exact) interior nodal values are 5/72 and 1/24. d) If instead you use 3 elements of constant length ($L^e = 1/3$) verify that the source vector for the second element is $\mathbf{C}^{eT} = [3 \quad 1]/24$ and that the (exact) interior nodal values are each 1/16.

17. Obtain a Galerkin solution of $y'' - 2y'x/g + 2y/g = g$, for $g = (x^2 + 1)$, on $0 \le x \le 1$ with the boundary conditions $y(0) = 2$, $y(1) = 5/3$.

18. For the differential equation in Problem 2.17 if we have one essential boundary condition of $y(0) = 1$ and one Neumann flux boundary condition of $dy/dx(1) = -4/3$ the exact solution is unchanged. Obtain a Galerkin finite element solution and compare it to the exact result.

19. The Couette steady flow velocity,u, of an incompressible viscous fluid between two parallel plates, with a constant pressure gradient of dP/dx, is: $\mu d^2u/dy^2 = dP/dx$, where $u(0) = 0$ and $u(H) = U_H$ describe a lower ($y = 0$) fixed plate and an upper plate ($y = -h$) moving with a speed of U_H. One can obtain a finite element solution for the interior nodal velocites across the fluid. The associated volume flow, per unit z thickness, is $Q = \int_0^H u(y)dy$. Describe how you would post-process the elements to obtain it.

20. For the above Couette flow would you expect to obtain the exact $u(y)$ and Q values for a) one quadratic element, b) two quadratic elements, c) two linear elements? Explain why.

21. Differences between Galerkin and least squares finite element procedures are _____: a) Galerkin allows integration by parts, b) least squares always yield symmetric algebraic equations, c) least squares requires C^1 continuity for second order differential equations, d) all of the above.

22. Repeat the graphical source assembly shown in Fig. 2.10.3 using three elements instead of two.

23. Burnett presents detailed studies of a symmetric bi-material bar with not-symmetrical boundary conditions. The total bar length is $L = 100\,cm$ with the center-most $20\,cm$ section being constructed out of copper ($k_c = 0.92\,cal/sec - cm - C$, $\rho = 8.5\,gm/cm^3$, $c_p = 0.092\,cal/gm - C$) while the other two ends of the bar are made of $40\,cm$ of steel ($k_c = 0.12\,cal/sec - cm - C$, $\rho = 7.8\,gm/cm^3$, $c_p = 0.11\,cal/gm - C$). The circular bar has a radius of $2\,cm$. Along its full length it convects to surrounding air at $U_\infty = 20\,C$. The end area at $x = 0$ receives a flux of $q = 0.1\,cal/sec - cm^2$ and the end at $x = 100$ has a temperature of $U_{100} = 0\,C$. Obtain: a) a steady state solution with 5 equal length elements, b) a transient solution assuming an initial condition of $U(x, 0) = 20\,C$. Note that the two end "thermal shocks" will require a finer mesh at the ends if the early time history is important. Would the steady state solution also need that mesh feature?

 © 2004 J.E. Akin. All rights reserved.

2.19 Bibliography

[1] Adams, R.A., *Sobolev Spaces*, New York: Academic Press (1975).
[2] Ainsworth, M. and Oden, J.T., *A Posteriori Error Estimation in Finite Element Analysis*, New York: John Wiley (2000).
[3] Akin, J.E., *Finite Elements for Analysis and Design*, London: Academic Press (1994).
[4] Axelsson, O. and Baker, V.A., *Finite Element Solution of Boundary Value Problems*, Philadelphia, PA: SIAM (2001).
[5] Aziz, A.K., *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, New York: Academic Press (1972).
[6] Bonnerot, R. and Jamet, P., "Numerical Computation of the Free Boundary for the Two-Dimensional Stefan Problem by Space-Time Finite Elements," *J. Computational Physics*, **25**, pp. 163−181 (1977).
[7] Bruch, J.C. Jr. and Zyvoloski, G., "Transient Two-Dimensional Heat Conduction Problems Solved by the Finite Element Method," *Int. J. Num. Meth. Eng.*, **8**, pp. 481−494 (1974).
[8] Buchanan, G.R., *Finite Element Analysis*, New York: McGraw-Hill (1995).
[9] Ciarlet, P.G., *The Finite Element Method for Elliptical Problems*, Philadelphia, PA: SIAM (2002).
[10] DeBoor, C., ed., *Mathematical Aspects of Finite Elements in Partial Differential Equations*, London: Academic Press (1974).
[11] Desai, C.S., *Elementary Finite Element Method*, Englewood Cliffs: Prentice-Hall (1979).
[12] Heinrich, J.C. and Pepper, D.W., *Intermediate Finite Element Method*, Philadelphia, PA: Taylor & Francis (1999).
[13] Huang, H.C. and Usmani, A.S., in *Finite Element Analysis for Heat Transfer*, London: Springer-Verlag (1994).
[14] Hughes, T.J.R., *The Finite Element Method*, Englewood Cliffs: Prentice-Hall (1987).
[15] Liusternik, L.A. and Sobolev, V.J., *Elements of Functional Analysis*, New York: Frederick Ungar (1961).
[16] Meier, D.L., "Multi-Dimensional Astrophysical Structural and Dynamical Analysis, Development of a Nonlinear Finite Element Approach," *Astrophysics J.*, **518**, pp. 788−813 (1999).
[17] Mitchell, A.R. and Wait, R., *The Finite Element Method in Partial Differential Equations*, London: John Wiley (1977).
[18] Nowinski, J.L., *Applications of Functional Analysis in Engineering*, New York: Plenum Press (1981).
[19] Oden, J.T. and Reddy, J.N., *An Introduction to the Mathematical Theory of Finite Elements*, New York: John Wiley (1976).

 © 2004 J.E. Akin. All rights reserved.

[20] Oden, J.T., *Applied Functional Analysis*, Englewood Cliffs: Prentice-Hall (1979).
[21] Oden, J.T. and Carey, G.F., *Finite Elements: Mathematical Aspects*, Prentice Hall (1983).
[22] Oden, J.T., "The Best FEM," *Finite Elements in Analysis and Design*, **7**, pp. 103−114 (1990).
[23] Reddy, J.N. and Gartling, D.K., *The Finite Element Method in Heat Transfer and Fluid Dynamics*, Boca Raton: CRC Press (2001).
[24] Strang, W.G. and Fix, G.J., *An Analysis of the Finite Element Method*, Englewood Cliffs: Prentice-Hall (1973).
[25] Szabo, B. and Babuska, I., *Finite Element Analysis*, New York: John Wiley (1991).
[26] Tezduyar, T.E. and Ganjoo, D.K., "Petrov-Galerkin Formulations with Weighting Functions Dependent Upon Spatial and Temporal Discretization," *Comp. Meth. Appl. Mech. Eng.*, **59**, pp. 47−71 (1986).
[27] 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).
[28] Zhu, J.Z. and Zienkiewicz, O.C., "Superconvergence Recovery Techniques and *A Posteriori* Error Estimators," *Int. J. Num. Meth. Eng.*, **30**, pp. 1321−1339 (1990).
[29] Zienkiewicz, O.C. and Morgan, K., *Finite Elements and Approximation*, Chichester: John Wiley (1983).
[30] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method*, 4th Edition, New York: McGraw-Hill (1991).
[31] Zienkiewicz, O.C. and Zhu, J.Z., "Superconvergent Patch Recovery Techniques and Adaptive Finite Element Refinement," *Comp. Meth. Appl. Mech. Eng.*, **101**, pp. 207−224 (1992).
[32] Zienkiewicz, O.C. and Zhu, J.Z., "The Superconvergent Patch Recovery and *a Posteriori* Error Estimates. Part 2: Error Estimates and Adaptivity," *Int. J. Num. Meth. Eng.*, **33**, pp. 1365−1382 (1992).

 © 2004 J.E. Akin. All rights reserved.