4.0.3 A 1D Example

Let's now illustrate the Galerkin method by solving a couple of specific examples. First consider the solution of the 1D Poisson equation

$$\frac{d^2u}{dx^2} = 1 \tag{4.59}$$

with boundary conditions

$$u(0) = 0, \quad u(1) = 1 \tag{4.60}$$

We'll break up our solution domain, the line from $[0, 1]$ into two elements and have three nodes at $x_1 = 0, x_2 = .5, x_3 = 1$. Since the boundary conditions at x_1 and x_3 are already specified, we're really just interested in find the solution of u at node x_2. Let $\epsilon_i = \Delta x_i = x_{i+1} - x_i$, such that $\Delta x_1 = x_2 - x_1$ and $\Delta x_2 = x_3 - x_2$. To approximate u at $x = x_2$ we need to solve the equation,

$$\int_0^1 \frac{d\Phi_i}{dx}\frac{d\Phi_j}{dx}dx = -\int_0^1 f(x)\Phi_j dx \tag{4.61}$$

The left side of the equation is evaluated as

$$\int_0^1 \frac{d\Phi_i}{dx}\frac{d\Phi_j}{dx}dx = \frac{1}{\Delta x_1} + \frac{1}{\Delta x_2} = \frac{\Delta x_1 + \Delta x_2}{\Delta x_1 \Delta_{x2}} \tag{4.62}$$

The right hand side of the equation is

$$\int_0^1 f(x)\Phi_j dx = \int_0^1 f(x)\Phi_1 dx + \int_0^1 f(x)\Phi_2 dx \tag{4.63}$$

or if we let $\xi = \frac{x - x_i}{x_{i+1} - x_i} = \frac{x - x_i}{\Delta x_i}$ and $dx = \Delta x_i d\xi$

$$\int_0^1 f(x)\Phi_j dx = \int_0^1 f(\xi)\xi d\xi + \int_0^1 f(\xi)(1-\xi)d\xi. \tag{4.64}$$

Since $f(x) = 1$, we have

$$\int_0^1 f(x)\Phi_j dx = \Delta x_1 \frac{\xi^2}{2}\Big|_0^1 + \Delta x_2 (1 - \frac{\xi^2}{2})\Big|_0^1 = \frac{\Delta x_1}{2} + \frac{\Delta x_2}{2} \tag{4.65}$$

Therefore we have

$$(\frac{\Delta x_1 + \Delta x_2}{\Delta x_1 \Delta_{x2}})u_2 = -(\frac{\Delta x_1}{2} + \frac{\Delta x_2}{2}) \tag{4.66}$$

or since $\Delta x_1 = x_2$ and $\Delta x_2 = 1 - x_2$ we have

$$u_2 = \frac{x_2(x_2 - 1)}{2} \tag{4.67}$$

The exact solutions is $u = \frac{x(x-1)}{2}$, so the Galerkin solution agrees exactly with the continuous solution.

Now let $f(x) = x$ with the same boundary conditions as before. All we need to do is re-evaluate the source term integrals

$$\int_{x_1}^{x_2} x\Phi_1 dx + \int_{x_2}^{1} x\Phi_2 dx = \int_{x_1}^{x_2} x(\frac{x-x_1}{x_2-x_1})dx + \int_{x_2}^{1} x(1-\frac{x-x_2}{x_3-x_2})dx \tag{4.68}$$

which yields

$$\frac{x_2^2}{3} + (\frac{x_2(1-x_2)}{2} + \frac{1}{6}(1-x_2)^2) \tag{4.69}$$

which gives us the finite element solution of

$$\frac{x_2(1-x_2)}{6} \tag{4.70}$$

which once again, agrees exactly with the continuous solution.

4.0.4 Example: Variational Solution of Possion's Equation

We let Ω_h consist of the set of functions such that our dependent variable is linear on each sub-interval I_j and is continuous on $[0,1]$. Furthermore, we insist that our test functions vanish at the boundary points, $v(0) = v(1) = 0$.

As parameters to describe $v \in \Omega_h$, we choose the values, $\alpha_j = v(x_j)$, at the node points x_j with $j = 0, 1, \ldots, N+1$. Now we let $\Phi_j \in \Omega_h$ be a set of *basis functions*:

$$\Phi_j(x) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.71}$$

thus, Φ_j is the continuous, piecewise linear functions that takes a value 1 at the node point, x_j and value 0 at other node points.

A function $v \in \Omega_h$ then can be represented as,

$$v(x) = \sum_{i=1}^{N} \alpha_i \Phi_i, \quad x \in [0,1] \tag{4.72}$$

This says that each $v \in \Omega_h$ can be written in a unique way as a *linear combination* of the basis functions Φ_i. We say then that Ω_h is a *linear space of dimension* N with *basis*, $\{\Phi\}_{i=1}^{N}$.

The finite element approximation for the differential equation can now be formulated as:

$$(\mathbf{V}) \text{ Find } u_h \in \Omega_h \text{ s.t. } (u_h', v') = (f, v) \quad \forall v \in \Omega_h \tag{4.73}$$

$$(\mathbf{M}) \text{ Find } u_h \in \Omega_h \text{ s.t. } F(u_h) \leq F(v) \quad \forall v \in \Omega_h \tag{4.74}$$

We note that the variational formulation, **V** is equivalent to the Galerkin method and the minimization formulation, $\underline{M}$ is equivalent to the Ritz method.

Let's proceed along with the variational or Galerkin method. Since $u_h \in \Omega_h$ satifies the previous equation, then so does,

$$(u_h', \Phi_j') = (f, \Phi_j) \tag{4.75}$$

and if these equations hold, then by taking linear combinations, we see that u_h satifies $(u_h', v') = (f, v)$. Since

$$u_h(x) = \sum_{i=1}^{N} \xi_i \Phi_i(x) \tag{4.76}$$

then we can rewrite $(u_h', \Phi_j') = (f, \Phi_j)$ as,

$$\sum_{i=1}^{N} \xi_i (\Phi_i', \Phi_j') = (f, \Phi_j) \quad j = 1 \ldots, N. \tag{4.77}$$

which again constitutes a linear system of equations with N equations and N unknowns (ξ_i). In matrix form, we can reformulate this as,

$$A\xi = b \tag{4.78}$$

4.0.5 Two-dimensional problems

Now that we have a feeling for how the finite element method works and have applied it to a simple 1D problem, let's know work up the machinery for handling 2D problems. We note that the extension to 3D is straightforward in terms of the theory and algorithms. The difficulties when extending to 3D (the so-call *curse of dimensionality*) have to do with the size of the resulting system and in creating the finite element mesh.

Let's start with an arbitrary 2D planar domain Ω and break it up into discrete elements to form a finite dimensional subspace, Ω_h. For 2D planar domains we have the choice of representing our function (if we assume for now that we will use linear elements) as either triangles,

$$u \approx \tilde{u} = \alpha_1 + \alpha_2 x + \alpha_3 y \tag{4.79}$$

or as quadralaterals,

$$u \approx \tilde{u} = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy \tag{4.80}$$

For now, let's restrict our development to triangles, knowing that it is easy to modify our formulas for quadralaterals. We take out a specific triangle from our finite dimensional subspace and apply the previous formulations for the three vertices,

$$\begin{aligned}
\tilde{u}_1 &= \alpha_1 + \alpha_2 x_1 + \alpha_3 y_1 \\
\tilde{u}_2 &= \alpha_1 + \alpha_2 x_2 + \alpha_3 y_2 \\
\tilde{u}_3 &= \alpha_1 + \alpha_2 x_3 + \alpha_3 y_3
\end{aligned} \tag{4.81}$$

or in matrix form,

$$\begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}. \tag{4.82}$$

We can now solve for the coefficients:

$$\alpha_1 = \frac{1}{2A} \begin{vmatrix} \tilde{u}_1 & x_1 & y_1 \\ \tilde{u}_2 & x_2 & y_2 \\ \tilde{u}_3 & x_3 & y_3 \end{vmatrix} \tag{4.83}$$

$$\alpha_2 = \frac{1}{2A} \begin{vmatrix} 1 & \tilde{u}_1 & y_1 \\ 1 & \tilde{u}_2 & y_2 \\ 1 & \tilde{u}_3 & y_3 \end{vmatrix} \tag{4.84}$$

$$\alpha_3 = \frac{1}{2A} \begin{vmatrix} 1 & x_1 & \tilde{u}_1 \\ 1 & x_2 & \tilde{u}_2 \\ 1 & x_3 & \tilde{u}_3 \end{vmatrix} \tag{4.85}$$

where

$$2A = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{4.86}$$

If we now define,

$$\begin{aligned} a_1 &= x_2 y_3 - x_3 y_2 \\ b_1 &= y_2 - y_3 \\ c_1 &= x_3 - x_2 \end{aligned} \tag{4.87}$$

we can express our coefficients as

$$\begin{aligned} \alpha_1 &= (a_1 \tilde{u}_1 + a_2 \tilde{u}_2 + a_3 \tilde{u}_3)/2A \\ \alpha_2 &= (b_1 \tilde{u}_1 + b_2 \tilde{u}_2 + b_3 \tilde{u}_3)/2A \\ \alpha_3 &= (c_1 \tilde{u}_1 + c_2 \tilde{u}_2 + c_3 \tilde{u}_3)/2A \end{aligned} \tag{4.88}$$

with

$$2A = a_i + b_i x_i + c_i y_i \tag{4.89}$$

were we can just permute the indicies $1, 2, 3$ to find a_2, b_2, etc. Now we can write an expression for a general point in our triangle in terms of the values at the vertices,

$$\tilde{u} = \frac{1}{2A}[(a_1 + b_1 x + c_1 y)\tilde{u}_1 + (a_2 + b_2 x + c_2 y)\tilde{u}_2 + (a_3 + b_3 x + c_3 y)\tilde{u}_3] \tag{4.90}$$

or, most succinctly,

$$\tilde{u} = \sum_i N_i \tilde{u}_i \tag{4.91}$$

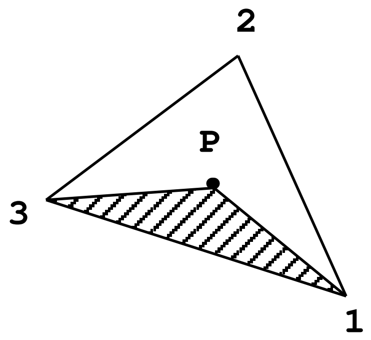

Figure 4.6: Coordinate transformation to barycentric coordinates.

with $\tilde{u}_i$ the solution value at node i and N_i is called the local *shape function* or *basis function.* This can be expressed in a variety of ways in the literature (depending, usually, on whether you are reading engineering or mathematical treatments of finite element analysis):

$$\Phi_j(N_i) = N_i(x, y) = f_i(x, y) \equiv \frac{a_i + b_i x + c_i y}{2A} \tag{4.92}$$

From our definition of $2A$ we can see that $N_1 = 1$ at node 1 and $N_1 = 0$ at nodes $2, 3$. In general this can be expressed as,

$$\Phi_j(N_i) \equiv \delta_{ij} \tag{4.93}$$

where δ_{ij} is the Kroneker delta.

Our expression for the shape functions is, in fact, a transformation from cartesian coordinates to a natural set of coordinates, often called *area coordinates* or *barycentric coordinates.* If we look at the transformation for N_2 for a point within the element with coordinates (x_p, y_p), we note that,

$$N_2 = \frac{a_2 + b_2 x_p + c_2 y_p}{2A} = \frac{Area(P13)}{Area(123)} \tag{4.94}$$

which is just a ratio of the areas of the triangles $(P13)$ and $(123) = A$ as shown in figure 4.6 Thus we have a transformation from a point inside an element in terms of its barycentric coordinates to the global cartesian system,

$$\begin{aligned} x &= N_1 x_1 + N_2 x_2 + N_3 x_3 \\ y &= N_1 y_1 + N_2 y_2 + N_3 y_3 \\ 1 &= N_1 + N_2 + N_3 \end{aligned} \tag{4.95}$$

Now that we have a suitable set of basis functions, we can find the finite element approximation to our 2D problem. Recall that our orginal problem can be formulated as,

$$a(u, v) = (f, v) \quad \forall v \in \Omega \tag{4.96}$$

where

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v dx \tag{4.97}$$

and

$$(f, v) = \int_\Omega f \cdot v dx \tag{4.98}$$

The finite element approximation to the original boundary value problem is

$$a(u_h, v) = (f, v) \quad \forall v \in \Omega_h \tag{4.99}$$

which has the equivalent form

$$\sum_{i=1}^{N} \xi_i a(\Phi_i, \Phi_j) = (f, \Phi_j) \tag{4.100}$$

where

$$a(\Phi_i, \Phi_j) = a(\Phi_i(N_j), \Phi_j(N_i)) \tag{4.101}$$

which can be expressed by the matrix and vector elements,

$$(a_{ij}) = \int_{\Omega_E} (\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}) d\Omega \tag{4.102}$$

and

$$f_i = \int_{\Omega_E} N_i f \, d\Omega \tag{4.103}$$

Fortunately, the above quantities are easy to evaluate for linear triangles. As a matter of fact, there are closed form solutions for the matrix elements (a_{ij}).

$$\int_{\Omega_h} N_1^a N_2^b N_3^c \, d\Omega = 2A \frac{a!b!c!}{(a+b+c+2)!} \tag{4.104}$$

therefore,

$$(a_{ij}) = \int_{\Omega_E} \frac{b_i b_j + c_i c_j}{4A^2} \, d\Omega = \frac{b_i b_j + c_i c_j}{4A} \tag{4.105}$$

We can compute the so-called, *first-order differentiation matrices* using the following algorithm:

Algorithm FE 1: Diff_Triangle(b, c, A, x, y)
Function: Computes the 1st order differentiation arrays for linear triangles
Output: **b** and **c**, the 1×3 differentiation arrays
Inputs: **x** and **y**, the 3×1 array of coordinates at the three vertices of the triangle.

Begin

Area2 $\leftarrow x[2]*y[3]-x[3]*y[2]+x[3]*y[1]$
- x[1]*y[3] + x[1]*y[2] - x[2]*y[1]

$A \leftarrow Abs(\text{Area2})/2$

For i $\leftarrow$ 1 to 3 Do

$i1 \leftarrow \text{mod}(i,3)+1$

$i2 \leftarrow \text{mod}(i1,3)+1$

$\mathbf{b}[i] \leftarrow (y[i1]-y[i2])/\text{Area2}$

$\mathbf{c}[i] \leftarrow (x[i2]-x[i1])/\text{Area2}$

End

The right hand side we have, assuming we have constant sources, can be calculated as

$$f_i = \int_{\Omega_E} \frac{a_i + b_i x + c_i y}{2A} f \, d\Omega = \frac{Af}{3}. \tag{4.106}$$

If we do now have $f(x,y) =$ constant, then we have to numerically approximate the integrals. For a single triangle, the local, so-called *element matrix* looks like

$$\frac{1}{4A}\begin{pmatrix} b_1^2+c_1^2 & b_1b_2+c_1c_2 & b_1b_3+c_1c_3 \\ & b_2^2+c_2^2 & b_2b_3+c_2c_3 \\ \text{Sym.} & & b_3^2+c_3^2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \frac{Af}{3}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \tag{4.107}$$

Which have the compact forms,

$$a_{ij}^{(n)} = \frac{1}{4A}(b_i^{(n)}b_j^{(n)} + c_i^{(n)}c_j^{(n)}) \tag{4.108}$$

and

$$f_i^{(n)} = \frac{Af}{3} \quad \text{for constant sources.} \tag{4.109}$$

We can compute the local matrices using the following algorithm:

Algorithm FE 2: Local_Matrices(A, f, x, y, source)

Function: Computes the local element matrices for linear triangles

Outputs: $\mathbf{A}$, the 3×3 local element matrix for a single triangle, $\mathbf{A} = (a_{ij})$. $\mathbf{f}$, the 3×1 local right hand side.

Inputs: $\mathbf{x}$ and $\mathbf{y}$, the 3×1 array of coordinates at the three vertices of the triangle, source = the 3×1 vector giving the source value at each node of the triangle (or a constant).

Requirements: Algorithm FE 1: Diff_Triangle

Begin

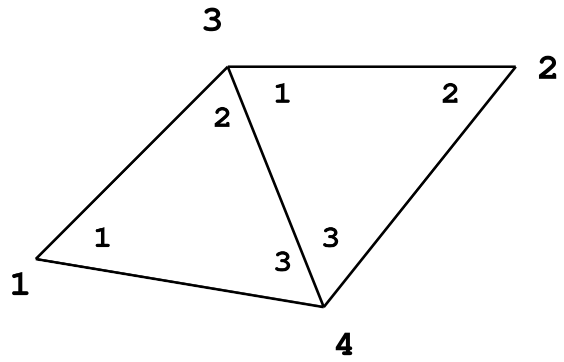

Figure 4.7: Adding two elements. Global numbers are on the outside;local are inside.

Diff_Triangle($\mathbf{b}$, $\mathbf{c}$, A, $\mathbf{x}$, $\mathbf{y}$)
For i $\leftarrow$ 1 to 3 Do
$f[i] \leftarrow A * (2 * \text{source}[i] + \text{source}[i1] + \text{source}[i2])/12$
(or if source = constant, replace with $f[i] \leftarrow A * \text{source}/3$
For j $\leftarrow$ 1 to 3 Do
$A[i,j] \leftarrow A * (\mathbf{b}[i] * \mathbf{b}[j] + \mathbf{c}[i] * \mathbf{c}[j])$

End

Now we must add up all the contributions from each element into a global matrix and global vector.

$$\sum_{n=1}^{Nel} (a_{ij}^{(n)})(\xi_i) = (f_i^{(n)}) \tag{4.110}$$

where Nel is equal to the total number of elements in the discretized solution domain and i represents the node numbers (vertices). Let's say we want to add the contributions from two elements which have global numbers (134) for element one and (324) for element two as shown in figure 4.7. The local element matrices look like

$$\begin{pmatrix} a_{11}^1 & a_{12}^1 & a_{13}^1 \\ & a_{22}^1 & a_{23}^1 \\ S & & a_{33}^1 \end{pmatrix} \tag{4.111}$$

and

$$\begin{pmatrix} a_{11}^2 & a_{12}^2 & a_{13}^2 \\ & a_{22}^2 & a_{23}^2 \\ S & & a_{33}^2 \end{pmatrix}. \tag{4.112}$$

Where the S denotes symmetry. The compenents of (a_{ij}) represent the connectivities in the global matrix and are the contributions from the (Φ_i, Φ_j) pair. Thus the two element system would have the following global matrix in terms of the local contributions.

$$\begin{pmatrix} a^1_{11} & 0 & a^1_{12} & a^1_{13} \\ & a^2_{22} & a^2_{12} & a^2_{23} \\ & & a^1_{22}+a^2_{11} & a^1_{23}+a^2_{13} \\ S & & & a^1_{33}+a^2_{33} \end{pmatrix}. \tag{4.113}$$

This process continues until we have added in all our element contributions to form the global system. The following algorithm creates the global matrix from the element matrices.

Algorithm FE 3: Global_Matrices($\mathbf{A}_g$, $\mathbf{f}_g$, A, f, vert, u)

Function: Creates the global matrices from the local matrices.

Outputs: $\mathbf{A}_g$ = the $N \times N$ global matrix from all the local matrix contributions; $\mathbf{f}_g$ = the $N \times 1$ global right hand side.

Inputs: $\mathbf{A}$ = the 3×3 local matrix, $\mathbf{f}$, the 3×1 local right hand side, N = number of unknown nodes, **vert** = the connectivities of the triangles; $\mathbf{u}$ = a vector equal to the entire number of nodes in the problem (number of unknowns, N, plus the number of known boundary nodes). Allow the first N elements to be unknown and those which we are solving for while the rest of the nodes are known due to boundary conditions.

Begin

```
For Row ← 1 to 3 Do
    If vert[Row] ≤ N
    Then ! Else Row corresponds to a known value so ignore it.
        fg[vert[Row]], ← fg[vert[Row]] + f[Row]
        For Col ← 1 to 3 Do
        If vert[Col] ≤ N
          Then ! Add to global matrix.
               Ag[vert[Row]vert[Col]] ← Ag[vert[Row]vert[Col]] +A[Row,Col]
          Else ! Column corresponds to a known value, shift right.
               fg[vert[Row]] ← fg[vert[Row]] - A[Row,Col]]*u[vert[Col]]
```

End

Now we just have to take into account of our boundary conditions and then solve our system.

4.0.6 Boundary Conditions

When we covered the finite difference method, you may remember that the Dirichlet boundary conditions were very easy to apply while the Neumann condition took a little extra effort.