Finite Element Treatment of the Navier Stokes Equations, Part II: Finite element geometry and basis functions

John Burkardt
https://people.sc.fsu.edu/~jburkardt/fem_2005/fem_ns2.pdf

15 August 2021

1 Introduction

This is part two of a set of short notes explaining how the finite element method was used to solve a class of flow problems governed by the Navier Stokes equations.

The first set of notes began with a fairly general version of the Navier Stokes equations, and derived the specific version that we are interested in, namely, the equations appropriate for steady-state incompressible flow in two space dimensions. It was noted that these equations are nonlinear, and a formal version of Newton's method was set up. It wasn't really possible to proceed beyond this formal structure, as long as we insist on the correct but intractable approach of treating the state variables as arbitrary functions with, in a sense, an infinite number of degrees of freedom.

Now we look at the employment of the theory of finite elements to develop a tractable problem from our intractable problem. Specifically, we are concerned with the idea of approximating the state functions by a set of simpler functions that can be described by a discrete set of data and the development of the simpler functions from finite element basis functions associated with a discretization of the problem geometry.

2 An Overview of the Finite Element Method

The finite element method offers us one way to start with a flow problem governed by the Navier-Stokes equations, and produce approximations to the flow whose accuracy can be increased as we desire.

There are a number of steps in implementing the finite element method. Before we jump into them, it will be helpful to survey the coming discussion.

Our procedure will begin by discretizing the problem. This means we will first discretize the geometry of the flow problem. This will allow us to develop a discretized representation for arbitrary pressure and velocity functions. We do this by building spaces of functions that are related to the discretized geometry. Essentially, each node of the mesh will have one or two basis functions associated with it, of a simple form.

Once we have a zoo of possible pressure and velocity functions, we need a procedure for telling us which is the "right" one, that is, a good approximation to the problem we are trying to solve. Since our functions can now be completely described by a small number of coefficients, we only need to determine these numbers.

We begin by applying our original Navier-Stokes equations to the presumed solution. We then set up the Galerkin procedure, which will allow us to generate one equation for each unknown coefficient. We note that the Galerkin formulation requires us to evaluate the integral of various quantities. Finally, we do some manipulation of the equations, using integration by parts, to lower the differentiability requirements on the velocity.

We now have a set of nonlinear equations for the coefficients, and we can set up the usual Newton iteration to seek a solution.

This still leaves open many questions, such as how we are going to evaluate the basis functions and how to evaluate the integrals required by the Galerkin procedure. These questions will be taken up when we get to the final sections on the **implementation** of the finite element method.

3 Discretizing the Geometry

The key to getting our problem solved is to discretize it. The first thing we need to work on is the geometry of the flow region Ω. Extra complications can occur if the boundary $\partial\Omega$ is curved, but we will start by assuming we're working in a rectangle.

The state equations are supposed to hold everywhere inside the region, and the boundary conditions everywhere along the boundary. In a finite difference or collocation setting, we would replace this requirement by one that specified the conditions to hold on a discrete mesh of nodes.

The finite element will seem to adopt this approach as well. We will place nodes along the boundary and throughout the region. We will see, however, that instead of requiring the equations to be satisfied at a

point, we will be, in a sense, averaging the behavior of the equations over little elements of area, which is a defining feature of the finite element approach.

So we now use a set of nodes to break up our region and its boundary, but we keep in mind the fact that we will want to carry out certain operations over the interior of the individual elements.

For our work, these elements will look like triangles. In fact, the technical name for them is *Taylor-Hood elements*. Instead of just the three vertices you would expect to use to define a triangle, we will include the midsides as nodes as well, giving us six nodes per triangle.

The technical reasons for the six nodes on the triangle are related to the fact that we have two variables to approximate, pressure and velocity. The linear approximation to the pressure will use just the vertices of the triangle, while the more accurate approximation of velocities will also require the midside nodes.

All the nodes will participate in approximating the velocity, but only some of them will be used for pressure. We call these nodes *pressure nodes*.

We will need at times to compute the area of a triangle T whose vertices are (x_i, y_i), (x_j, y_j) and (x_k, y_k). This can be expressed compactly as

$$Area(T) = \frac{1}{2}\, det \begin{bmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{bmatrix} \tag{1}$$

We will also refer to a similar formula for the volume of a tetrahedron τ:

$$Volume(\tau) = \frac{1}{6}\, det \begin{bmatrix} x_i & y_i & z_i & 1 \\ x_j & y_j & z_j & 1 \\ x_k & y_k & z_k & 1 \\ x_l & y_l & z_l & 1 \end{bmatrix} \tag{2}$$

4 The Linear Basis Functions for Pressure

The solution to the Navier Stokes equations is a pair of functions $\mathbf{v}(x, y)$ and $p(x, y)$. Since we can't hope to determine these functions directly, we seek ways of approximating them, using discrete methods. We'll start with the slightly simpler question of approximating the scalar pressure function.

Rather than looking for a function defined at all points in the flow region, we will imagine our pressure function is known only at a few points in the region; we will not be able to determine what the pressure function "really" does elsewhere. Instead, between the known points, we will have some kind of model of the function's behavior.

(This is an important distinction, since in a finite difference approach, for instance, no attempt is made to describe the behavior of the function except at the sample points. The finite element method, in contrast, creates a model or approximation function to the unknown solution.)

One of the simplest models for the pressure function's behavior is to assume that we know its value at the three corner nodes (or "vertices") of each triangular element. This subset of the vertices will be called the "pressure nodes". While we will concentrate on the value of the function at these nodes, we will include a model of the function's behavior at other places, namely, that for any point within that element, the pressure is the linear interpolant of the values at the three vertices.

We can extend this same approach to every element; the function is not defined outside of the triangulated region. A little though will convince you that, along the boundaries shared by two elements, the function will be well defined even though it can be evaluated using one element or the other. Hence, we have now defined a function over the entire area covered by the elements, but which is describable by a small set of values. This function is continuous everywhere. It is not quite differentiable everywhere, but the nondifferentiability is a very mild kind, occurring only at the "seams" where one element meets another.

(Since, in fact, the pressure is differentiated in the Navier-Stokes equations, it is appropriate to ask whether it is appropriate to represent the pressure by a function which, in fact, cannot be differentiated. We will see soon that this is not a problem.)

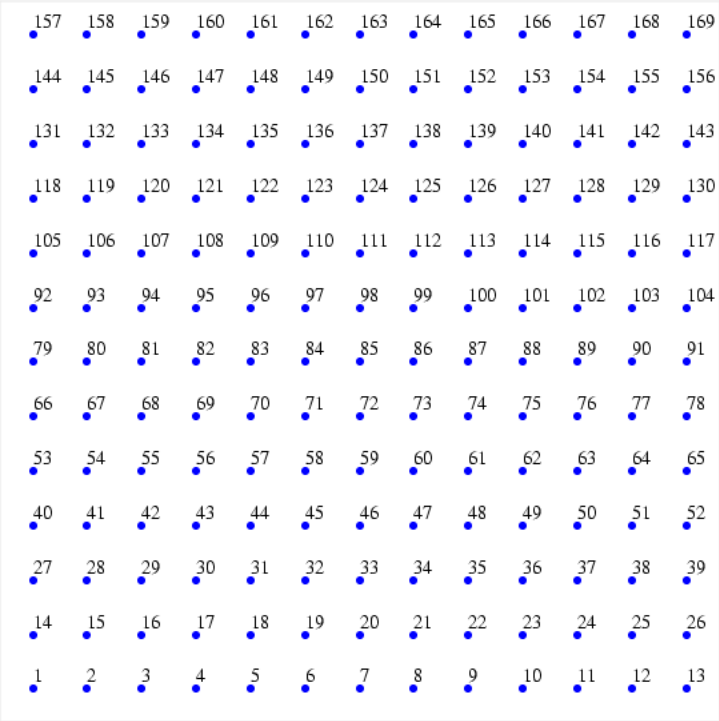

Figure 1: The flow region, filled with a grid of nodes.

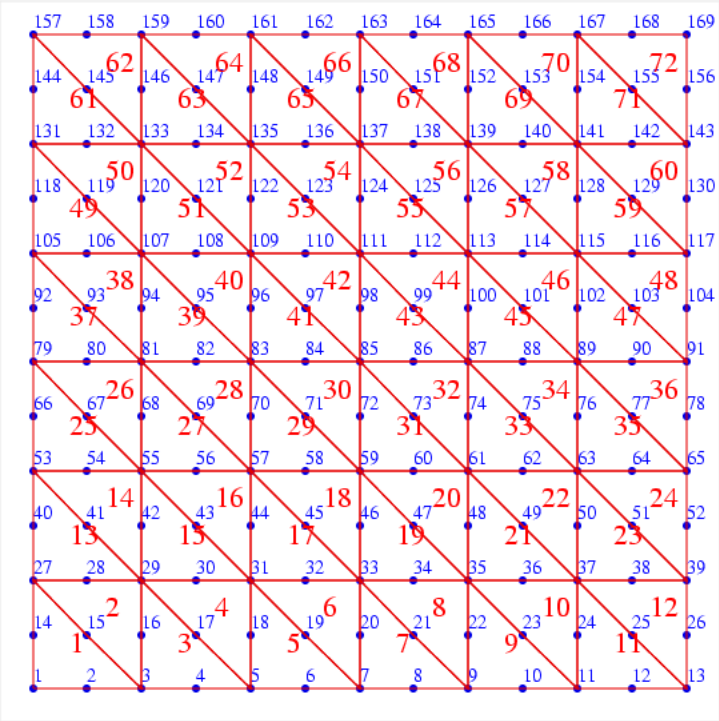

Figure 2: The subdivision of the flow region into Taylor-Hood elements.

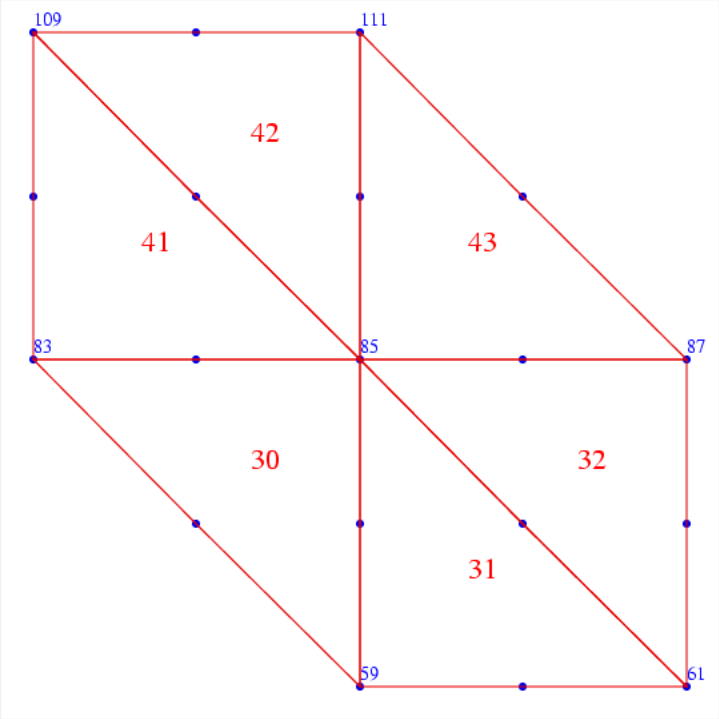

Figure 3: The support of the linear basis function associated with node 85.

As soon as we pick a set of elements for a flow region, we now have an associated space of piecewise linear functions that contain all possible candidates for the pressure function. We have already said that a candidate function for pressure can be completely described by its values at the pressure nodes. A simple basis for this space of candidate pressure functions is the set of functions $\{\phi_i(x,y) |$ i indexes a pressure node$\}$. The function $\phi_i(x,y)$ is 1 at pressure node i, and zero at all other pressure nodes. (Note that we are *not* specifying the value of ϕ_i at nodes that aren't pressure nodes.)

A given node is usually shared by two elements (if it is a midside node), or by many elements, (often six in our case) if it is a vertex. Thus, when a basis function is nonzero at some node i, this implies that the basis function is nonzero over every triangle for which node i is a vertex. The support of a basis function is precisely those elements which include the associated node. Over this region, the piecewise linear basis function looks something like a tent, with a single "pole" at 1, and flat tent-like surfaces dropping down to zero at the far side of each element.

For a simple, regular mesh, then, it might be the case that a linear basis function is nonzero over six elements; in more general cases, a basis function might be nonzero over three, or over forty elements, depending on the geometry we have set up. This makes it a bit tricky to try to take the perspective of a particular basis function, and try to discover all the elements it is involved with. On the other hand, if we look at things from the perspective of a particular element, there will *always* be just three nonzero piecewise linear basis functions over that element, and these are easy to store in a simple array.

This is one reason why, in a finite element program, when we try to gather up all the nonzero basis functions values, we first loop over elements, and then loop over the fixed number of basis functions which are nonzero there. Doing the loops in the opposite order would be much more difficult!

Now it should be clear that if we wish to create a discretized represention of a pressure function over our triangulated region, we need only know its values $pval_i$ at every pressure node. Supposing there are Np

pressure nodes, then a formula for the pressure function could be written as

$$p^h(x,y) = \sum_1^{Np} pval_i \; \phi_i(x,y) \tag{3}$$

Here, we use a superscript in p^h to indicate a function in our discretized space. The h is meant to suggest the size of the largest element, and thus functions as a sort of "index" when we are ready to look at limits.

Now the real key to the finite element representation is these basis functions. We know that the formula for a pressure basis function is a piecewise linear function; that the "pieces" correspond to elements; that the basis function is zero over most elements; that it is nonzero only in elements that include the (pressure) node associated with the basis function; and that the basis function is 1 at its associated node, and zero at all other (pressure) nodes.

Actually, this is enough for us to work out the formula for a pressure basis function. To see this, consider the pressure basis function $\phi_i(x,y)$, associated with pressure node n_i. Since the basis function is defined piece by piece, let's assume we are interested in the formula which applies in element $\mathbf{e}$, whose other two vertices we will call n_j and n_k.

We know that $\phi_i(x,y) = 0$ at nodes n_j and n_k. Therefore, being a linear function, it must also be zero at all points (x,y) on the line between these two nodes. But since these points lie on a line, we already know one linear relationship they satisfy:

$$\frac{y_k - y_j}{x_k - x_j} = \frac{y - y_j}{x - x_j} \tag{4}$$

(It's customary to write the slope relationship this way. Should either denominator be zero, we could eliminate the fractions, and have a valid, if less familiar, formula.)

If we subtract one side from the other, and call the result $g(x,y)$, we have that:

$$g(x,y) = (x - x_j)(y_k - y_j) - (x_k - x_j)(y - y_j) \tag{5}$$

We know that $g(x,y) = 0$ for those points on the line between n_j and n_k. Assuming our triangle is not degenerate, then $g(x_i,y_i)$ is *nonzero* (because n_i does *not* lie on the line between n_j and n_k!). So $g(x,y)$ is almost a basis function, since it's zero at the right places, and nonzero at the other. But it's easy to scale a function so that a nonzero value is 1. How's this for a candidate for our basis function:

$$\phi_i(x,y) = \frac{g(x,y)}{g(x_i,y_i)} \tag{6}$$

You should easily see that this formula is indeed zero at n_j and n_k, and sure enough, it's 1 at n_i, and therefore, it represents our basis function. Of course, it would be nice to see this formula explicitly. All we have to do is substitute, to get:

$$\phi_i(x,y) = \frac{(x - x_j)(y_k - y_j) - (x_k - x_j)(y - y_j)}{(x_i - x_j)(y_k - y_j) - (x_k - x_j)(y_i - y_j)} \tag{7}$$

This is a linear function of two arguments. Putting in the values (x_i,y_i), (x_j,y_j), and (x_k,y_k) gives you the values 1, 0 and 0, respectively, so we know it's right.

Interestingly, the value on the bottom of the fraction is essentially the area of the triangle (it's been multiplied by 2). If you look at this expression, you will start to see that the value of basis function $\phi_i(x,y)$ is the *relative area* of the triangle formed by (x,y) with the two base nodes (x_j,y_j) and (x_k,y_k). Note also that this value is constant along any line parallel to the base line. So you should now be able to easily sketch lines of constant $\phi_i(x,y)$ in a sample triangle.

Here's another beautiful thing about these functions. What is the sum $\phi_i(x,y) + \phi_j(x,y) + \phi_k(x,y)$? Well, plot the point in the big triangle. The value of $\phi_i(x,y)$ is the relative area of the triangle formed by (x,y) and the side of the triangle opposite to node n_i. Draw that triangle. Now draw the other two triangles

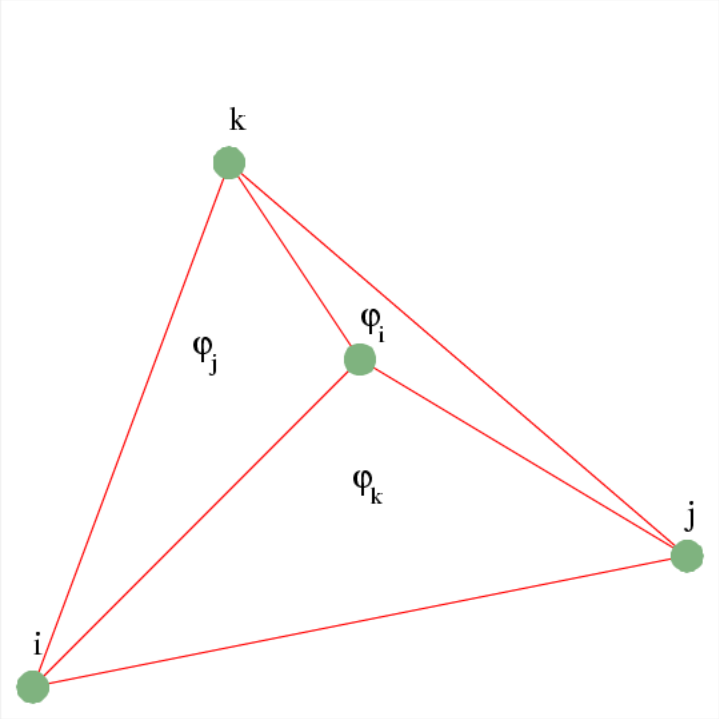

Figure 4: The value of each basis function is the relative area of the associated subtriangle!

associated with $\phi_j(x,y)$ and $\phi_k(x,y)$. You have just decomposed the full triangle into three subtriangles. No matter where you put (x,y) in the triangle, the area of the three subtriangles must sum to the total area, so the sum of the basis functions is 1!

This interesting fact about the three basis functions is related to what is known as the *barycentric coordinate system* for a triangle. Essentially, you can replace the coordinates (x,y) of a point in a triangle by the coordinates $(\phi_i(x,y), \phi_j(x,y), \phi_k(x,y))$. (Since the three barycentric coordinates always add up to 1, it is common to omit the third one.) Using this coordinate system makes it easy to find corresponding points in two triangles, assuming that a linear map is used which takes the three vertices of one triangle to the other. In particular, this is an alternative way to relate the physical and reference triangles.

Finally, we note that the integral of any linear basis function over the element is equal to 1/3 of the area of the element. This should be plausible, at least, since

$$\int_T \phi_i(x,y) + \phi_j(x,y) + \phi_k(x,y)\, dx\, dy = \int_T 1\, dx\, dy = Area(T) \tag{8}$$

To see that basis function $\phi_i(x,y)$ separately integrates to 1/3 the area, just realize that a 3D graph of the basis function has the shape of a tetrahedron τ, formed by the triangle T and the point $(x_i, y_i, 1)$:

$$Volume(\tau) = \frac{1}{6}\, det \begin{pmatrix} x_i & y_i & 0 & 1 \\ x_j & y_j & 0 & 1 \\ x_k & y_k & 0 & 1 \\ x_i & y_i & 1 & 1 \end{pmatrix} \tag{9}$$

Now expand in terms of column 3:

$$Volume(\tau) = \frac{1}{6}\, det \begin{pmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{pmatrix} = \frac{1}{3}\, Area(T) \tag{10}$$

So the integral of each linear basis function is 1/3 the area of the triangle, no matter how distorted the triangle is!

That's a fairly long explanation for a simple formula. But this should help you to see the reasoning and tricks that are used to come up with such formulas.

5 Linear Basis Function Routine

Here is a sample routine, written in FORTRAN90, which evaluates the linear basis functions associated with the vertices of a triangle.

```
subroutine basis_linear ( t, i, p, qi, dqidx, dqidy )

!*****************************************************************************
!
!! BASIS_LINEAR evaluates a linear basis function.
!
!  Discussion:
!
!    The routine is given the coordinates of the nodes of a triangle.
!
!           3
!          / \
!         /   \
!        /     \
!       1-------2
!
!    It evaluates the linear basis function Q(I)(X,Y) associated with
!    node I, which has the property that it is a linear function
!    which is 1 at node I and zero at the other two nodes.
!
!  Modified:
!
!    17 June 2005
!
!  Author:
!
!    John Burkardt
!
!  Parameters:
!
!    Input, real T(2,3), the coordinates of the nodes.
!
!    Input, integer I, the index of the desired basis function.
!    I should be between 1 and 3.
!
!    Input, real P(2), the coordinates of a point at which the basis
!    function is to be evaluated.
!
!    Output, real QI, DQIDX, DQIDY, the values of the basis function
!    and its X and Y derivatives.
!
```

```
  implicit none

  real ( kind = 8 ) area
  real ( kind = 8 ) dqidx
  real ( kind = 8 ) dqidy
  integer i
  integer i_wrap
  integer ip1
  integer ip2
  real ( kind = 8 ) p(2)
  real ( kind = 8 ) qi
  real ( kind = 8 ) t(2,3)

  area = t(1,1) * ( t(2,2) - t(2,3) ) &
       + t(1,2) * ( t(2,3) - t(1,1) ) &
       + t(1,3) * ( t(2,1) - t(2,2) )
!
!  IP1 and IP2 are the next two node indices in the sequence 1, 2, 3,
!  with wraparound.
!
  ip1 = i_wrap ( i + 1, 1, 3 )
  ip2 = i_wrap ( i + 2, 1, 3 )

  qi = ( ( t(1,ip2) - t(1,ip1) ) * ( p(2) - t(2,ip1) ) &
       - ( t(2,ip2) - t(2,ip1) ) * ( p(1) - t(1,ip1) ) ) / area

  dqidx = - ( t(2,ip2) - t(2,ip1) ) / area
  dqidy =   ( t(1,ip2) - t(1,ip1) ) / area

  return
end
```

6 The Quadratic Basis Functions for Velocity

You may already be concerned that the velocity is going to be more difficult to represent, since it is a vector quantity. However, this turns out to be easily handled, since any description of the velocity $\mathbf{v}$ is completely equivalent to a description of the scalar pair of functions (u, v), the horizontal and vertical velocity components.

So you can see the shape of the argument. We're going to show that there is an appropriate set of basis functions $\{\psi_i\}$ associated with our geometry; we'll show that the horizontal velocity function u can be approximated by a linear combination of these basis functions. It should be clear that the same set of basis functions can be used to approximate v, although with different linear coefficients, of course. Putting this all together gives us an approximation for $\mathbf{v}$.

We have already said that, for technical reasons, the Taylor Hood finite element will use a higher order approximation method for the velocities. In fact, when the pressure is approximated by piecewise linear functions, the velocity must be approximated by piecewise *quadratics*.

Now let's look at a single element. The reason we adorned all our elements with six nodes was precisely so that it could handle the case of quadratic functions. It takes a little thought to see that any function which is quadratic over such an element can be determined entirely by its values at the six nodes.

Suppose, then, that we specify a value of u at every node in the entire region Ω. We assume that these values represent a function; we'd like this function to be as smooth as possible inside each element. Since the function is defined at six points in the element, this means it must be a quadratic. (Six degrees of freedom can be represented by the functions $1, x, y, x^2, xy, y^2$). It turns out that if two elements share a side, they will both agree on the value of the quadratic function along their common side, simply because the function value along a side only depends on the nodal values along that side. However, the function will not be continuously differentiable at these element boundaries.

Our scheme, then, allows us to generate a space of piecewise quadratic functions. Again, it is natural to look for a suitable basis for this space, and simplicity suggests that, with each node i, we associate the basis function $\psi_i(x, y)$, which has the value 1 at that node, and which is 0 at all other nodes.

Using this basis, we may write any piecewise quadratic function as a linear combination of basis functions, so the discretized horizontal velocity component could be written as

$$u^h(x, y) = \sum_1^N uval_i \; \psi_i(x, y) \tag{11}$$

A similar formula would apply for $v^h(x, y)$. If we need to represent the vector velocity function, we have only to use vector-valued coefficients:

$$\mathbf{v}^h(x, y) = \sum_1^N \mathbf{vval}_i \; \psi_i(x, y) \tag{12}$$

A little thought should convince you that this does define a vector-valued function, that this is the right way to define a vector function, and that the two components are indeed fully independent, which must be the case if we want to be able to represent vector functions.

7 Formulas for Quadratic Basis Functions

It is common to introduce reference triangles in order to define basis functions. We did not do this in the linear case, and we will also avoid it for the case of quadratic basis functions, preferring to work, as far as possible, in the geometry of the individual elements. (Should we be interested in elements of higher degree, or elements in which the edges are allowed to curve, we would find a reference element more attractive!)

To define the quadratic basis functions, we will reason in a similar way to what we did for the linear basis functions. It turns out we only need to consider two cases: when the basis function is associated with a vertex (corner node) or a midside node.

So suppose that $\psi_i(x, y)$ is associated with a vertex. Then the two opposite nodes n_j and n_k can be described by a linear function, $g(x, y) = 0$ which is once again

$$g(x, y) = (x - x_j)(y_k - y_j) - (x_k - x_j)(y - y_j) \tag{13}$$

Moreover, by linearity, the midside node n_{jk} also satisfies $g(x, y) = 0$. Now we have two midside nodes remaining, which we call n_{ij} and n_{ik}. Since they lie on a line, they satisfy a linear relationship $h(x, y) = 0$; a little thought will convince you that this function is

$$h(x, y) = (x - x_{ij})(y_{ik} - y_{ij}) - (x_{ik} - x_{ij})(y - y_{ij}) \tag{14}$$

Finally, the node n_i satisfies neither relationship, that is, $g(n_i) \neq 0$ and $h(n_i) \neq 0$. So what do you think of the function $g(x, y)h(x, y)$? It is nonzero at n_i, and it is zero at the other five nodes in the element. Oh, and it is quadratic, of course. What more can we ask? We just need it to have the value 1 at n_i, and thus we can define

$$\psi_i(x, y) = \frac{g(x, y)\, h(x, y)}{(g(n_i)\, h(n_i))} \tag{15}$$

which we will *not* write out!

Now the case for any midside node is similar. Let us consider the basis function for n_{ij} then. Although we have now gotten into the habit of writing $\psi_i(x,y)$ as a generic basis function, we must, for clarity, write this desired basis function as $\psi_{ij}(x,y)$!

Now, as before, the nodes n_i and n_k lie on a straight line, so they satisfy a linear relation $g(x,y) = 0$ with

$$g(x,y) = (x - x_k)(y_i - y_k) - (x_i - x_k)(y - y_k) \tag{16}$$

Similarly, nodes n_j and n_k lie on a straight line, and satisfy a linear relation $h(x,y) = 0$ with

$$h(x,y) = (x - x_k)(y_j - y_k) - (x_j - x_k)(y - y_k) \tag{17}$$

Moreover, by linearity, the remaining midside nodes must satisfy $g(n_{ik}) = 0$ and $h(n_{jk}) = 0$. Unless the triangle is degenerate, neither function can be zero at n_{ij}. Therefore, we may confidently assert that the basis function we are seeking has the form:

$$\psi_{ij}(x,y) = \frac{g(x,y)\,h(x,y)}{(g(n_{ij})\,h(n_{ij}))} \tag{18}$$

(Note that the functions g and h are different from the previous case; they just represent two linear relations satisfied by the other nodes.)

Although our functions are more complicated than in the linear case, it will still be true that they add up to 1 at any point, that is

$$\psi_i(x,y) + \psi_j(x,y) + \psi_k(x,y) + \psi_{ij}(x,y) + \psi_{jk}(x,y) + \psi_{kj}(x,y) = 1 \tag{19}$$

This is one of the easiest ways to check if you have written your basis function routine correctly!

A way to see why this equation must be true is that the function $f(x,y) = 1$ can be represented using our basis; it has the value 1 at each node, so the coefficients will all be 1. Our statement above can then be regarded as another way of saying the function $f(x,y)$, even when represented as a sum of our basis functions, must be 1 everywhere (or at least within the element where we have defined the basis functions!).

Now you should see that it is relatively easy to evaluate either linear or quadratic basis functions in an element, using directly the geometric information represented by the positions of the nodes. For our application, it is *not* necessary to define a reference or master element; it is not necessary to determine forward and inverse mappings to each of the physical elements.

8 Quadratic Basis Function Routine

Here is a sample routine for computing the quadratic basis functions. Note that it has a calling sequence similar to that used by the linear basis function routine.

```
subroutine basis_quadratic ( t, i, p, bi, dbidx, dbidy )

!*****************************************************************************
!
!! BASIS_QUADRATIC evaluates a quadratic basis function.
!
!  Discussion:
!
!    The routine is given the coordinates of the nodes of a triangle.
!
!           3
!          / \
```

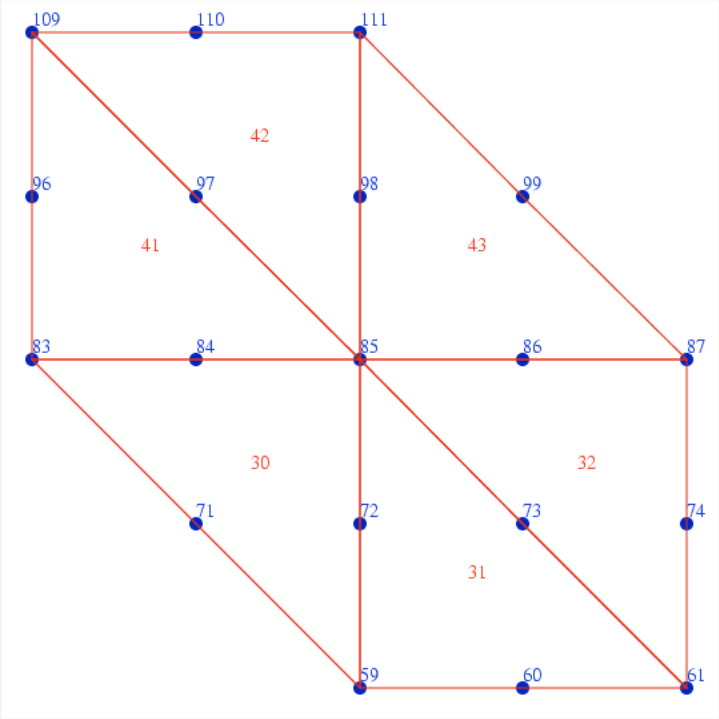

Figure 5: The support of the quadratic basis function associated with node 85.

```
!         6   5
!        /     \
!       1---4---2
!
!    It evaluates the quadratic basis function B(I)(X,Y) associated with
!    node I, which has the property that it is a quadratic function
!    which is 1 at node I and zero at the other five nodes.
!
!    This routine assumes that the sides of the triangle are straight,
!    so that the midside nodes fall on the line between two vertices.
!
!    This routine relies on the fact that each basis function can be
!    written as the product of two linear factors, which are easily
!    computed and normalized.
!
!  Modified:
!
!    17 June 2005
!
!  Author:
!
!    John Burkardt
!
!  Parameters:
!
```

```
!    Input, real T(2,6), the coordinates of the nodes.
!
!    Input, integer I, the index of the desired basis function.
!    T should be between 1 and 6.
!
!    Input, real P(2), the coordinates of a point at which the basis
!    function is to be evaluated.
!
!    Output, real BI, DBIDX, DBIDY, the values of the basis function
!    and its X and Y derivatives.
!
  implicit none

  real ( kind = 8 ) area
  real ( kind = 8 ) bi
  real ( kind = 8 ) dbidx
  real ( kind = 8 ) dbidy
  real ( kind = 8 ) gf
  real ( kind = 8 ) gn
  real ( kind = 8 ) hf
  real ( kind = 8 ) hn
  integer i
  integer i_wrap
  integer j1
  integer j2
  integer k1
  integer k2
  real ( kind = 8 ) p(2)
  real ( kind = 8 ) t(2,6)
!
!  Determine the pairs of nodes.
!
  if ( i <= 3 ) then
    j1 = i_wrap ( i + 1, 1, 3 )
    j2 = i_wrap ( i + 2, 1, 3 )
    k1 = i + 3
    k2 = i_wrap ( i + 5, 4, 6 )
  else
    j1 = i - 3
    j2 = i_wrap ( i - 3 + 2, 1, 3 )
    k1 = i_wrap ( i - 3 + 1, 1, 3 )
    k2 = i_wrap ( i - 3 + 2, 1, 3 )
  end if
!
!  Evaluate the two linear factors GF and HF,
!  and their normalizers GN and HN.
!
  gf = ( p(1)    - t(1,j1) ) * ( t(2,j2) - t(2,j1) ) &
     - ( t(1,j2) - t(1,j1) ) * ( p(2)    - t(2,j1) )

  gn = ( t(1,i)  - t(1,j1) ) * ( t(2,j2) - t(2,j1) ) &
```

```
     - ( t(1,j2) - t(1,j1) ) * ( t(2,i)  - t(2,j1) )

  hf = ( p(1)    - t(1,k1) ) * ( t(2,k2) - t(2,k1) ) &
     - ( t(1,k2) - t(1,k1) ) * ( p(2)    - t(2,k1) )

  hn = ( t(1,i)  - t(1,k1) ) * ( t(2,k2) - t(2,k1) ) &
     - ( t(1,k2) - t(1,k1) ) * ( t(2,i)  - t(2,k1) )
!
!  Construct the basis function and its derivatives.
!
  bi =        ( gf                  / gn ) * (   hf                  / hn )

  dbidx =   ( ( t(2,j2) - t(2,j1) ) / gn ) * (   hf                  / hn ) &
          + (   gf                  / gn ) * ( ( t(2,k2) - t(2,k1) ) / hn )

  dbidy = - ( ( t(1,j2) - t(1,j1) ) / gn ) * (   hf                  / hn ) &
          - (   gf                  / gn ) * ( ( t(1,k2) - t(1,k1) ) / hn )

  return
end
```

9 Mapping the Quadrature Rule

We will shortly find one relatively small cost of our choice to do business directly in the physical element: when we want to estimate integrals over the element, we will need to map the points of our quadrature rule into the element. That's because the points of a quadrature rule are always defined on a reference element. Typically, the rule for a triangle will be defined on a reference triangle Δ with vertices (0,0), (1,0) and (0,1).

Now the simplest quadrature rule of all on such a reference triangle is a one point rule. The single abscissa is at the centroid (1/3,1/3), the weight is $\frac{1}{2}$, and so the approximation to an integral over the reference triangle is

$$\int_{\Delta} f(\xi,\eta)\, d\xi\, d\eta \approx \frac{1}{2} f(\frac{1}{3},\frac{1}{3}). \tag{20}$$

Normally our quadrature rule would be more complicated. Let's try, however, to work out a systematic way of producing a corresponding quadrature rule for an arbitrary triangular element which we will call T. If we watch carefully what we do, we will be able to automatically handle more complicated quadrature rules.

First, let us assume that the midside nodes really are midway along the sides. This is natural for you to assume, but in more general cases, this might not be true, and it would invalidate our discussion, or force us to be more technical than we want to be. But as long as the midside nodes don't complicate our life, the mapping M from Δ to T will be linear – actually, M is *affine*, which is the next best thing to linear.

Let's settle some terminology now. We will label a typical point in Δ with the coordinates (ξ,η). The vertices of the triangle T will be indicated as n_i, n_j, and n_k. We assume that these are the images, respectively, of the reference triangle vertices (0,0), (1,0), and (0,1).

The mapping must have the form:

$$\begin{aligned} M(\xi,\eta) &= (a\xi + b\eta + c)\, n_i & (21)\\ &+ (d\xi + e\eta + f)\, n_j & (22)\\ &+ (g\xi + h\eta + i)\, n_k & (23) \end{aligned}$$

The mapping is affine because we cannot presume that c, f and i are 0.

With a little thought, we can work out the coefficients of M. Since (0,0) maps to n_i, we must have $c = 1$ and $f = i = 0$. Moroever, as long as η is 0, we only get points on the line between n_i and n_j. Therefore, g must be zero. Similar reasoning shows that e must be zero.

So now we know the mapping has this form:

$$\begin{aligned} M(\xi,\eta) &= (a\xi + b\eta + 1)\, n_i && (24)\\ &+ (d\xi)\, n_j && (25)\\ &+ (h\eta)\, n_k && (26) \end{aligned}$$

But since M(1,0) = n_j, we must have $a = -1$ and $d = 1$. Similarly, M(0,1) = n_k implies that $b = -1$ and $h = 1$.

Thus, we now have the complete form of our mapping:

$$M(\xi,\eta) = (1 - \xi - \eta)n_i + \xi n_j + \eta n_k \tag{27}$$

This result tells us how *any* point is to be linearly mapped. So in particular, the quadrature point (1/3,1/3) must map to

$$M(\xi,\eta) = \frac{1}{3}(n_i + n_j + n_k). \tag{28}$$

and that's how we transform the abscissa of the quadrature rule.

Of course, we also need to adjust the weight. The weight for the reference triangle is 1/2; that's because the area of the reference triangle is 1/2, and if we integrate the function 1, we must get the area. This is a general rule that the sum of the quadrature weights should equal the area of the region.

For general mappings, we would need to adjust each weight separately, using the Jacobian of the mapping. But a linear mapping has a constant Jacobian, which is simply the ratio of the image area to the original area. In other words, to adjust the weight, we divide it by 1/2 and multiply by the area of the triangle T.

The area of triangle T is

$$\begin{aligned} Area(T) &= \frac{1}{2}((x_i - x_j)(y_k - y_j) - (x_k - x_j)(y_i - y_j)) && (29)\\ &= \frac{1}{2}(x_i(y_k - y_j) + x_j(y_i - y_k) + x_k(y_j - y_i)) && (30) \end{aligned}$$

See also the earlier definition of area in terms of determinants.

Thus, our quadrature rule for the physical triangle T has become

$$\int_T f(x,y)\, dx\, dy \approx \frac{1}{2}\,(\frac{Area(T)}{1/2})\, f(\frac{1}{3}(n_i + n_j + n_k)). \tag{31}$$

A more accurate quadrature rule of order $n = 3$ is described by the following data:

Table 1:

i	ξ_i	η_i	w_i
1	$\frac{1}{2}$	0	$\frac{1}{6}$
2	0	$\frac{1}{2}$	$\frac{1}{6}$
3	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{6}$

The quadrature rule for the reference triangle Δ is then:

$$\int_\Delta f(\xi,\eta)\, d\xi\, d\eta \approx \sum_{i=1}^{n} w_i\, f(\xi_i,\eta_i). \tag{32}$$

Can you determine the form of this rule for the triangle T whose vertices are (1,1), (3,1) and (2,4)?

Since we worked out the transformation from the reference element to the physical element, it might be worth noting how to go the other way. First, rewrite the transformation from the reference element to the physical element in the matrix-vector form:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} (x_j - x_i) & (x_k - x_i) \\ (y_j - y_i) & (y_k - y_i) \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} \tag{33}$$

This immediately suggests how to compute the inverse transformation.

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \frac{1}{det(A)} \begin{pmatrix} (y_k - y_i) & -(x_k - x_i) \\ -(y_j - y_i) & (x_j - x_i) \end{pmatrix} \begin{pmatrix} x - x_i \\ y - y_i \end{pmatrix} \tag{34}$$

where, of course,

$$det(A) = (x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i) \tag{35}$$

10 Reference-to-Physical Mapping Routine

Here is a sample routine to map points in a reference triangle to points in a physical triangle. This is the sort of routine that can be used to transform the abscissas of a quadrature rule defined in a reference element.

```
subroutine reference_to_physical_t3 ( xn, yn, xsi, eta, x, y )

!*******************************************************************************
!
!! REFERENCE_TO_PHYSICAL_T3 maps a reference point to a physical point.
!
!  Discussion:
!
!    Given the vertices of an order 3 physical triangle and a point
!    (XSI,ETA) in the reference triangle, the routine computes the value
!    of the corresponding image point (X,Y) in physical space.
!
!    Note that this routine may also be appropriate for an order 6
!    triangle, if the mapping between reference and physical space
!    is linear.  This implies, in particular, that the sides of the
!    image triangle are straight and that the "midside" nodes in the
!    physical triangle are literally halfway along the sides of
!    the physical triangle.
!
!  Reference Element T3:
!
!    |
!    1  3
!    |  |\
!    |  | \
!    S  |  \
!    |  |   \
!    |  |    \
!    0  1-----2
!    |
!    +--0--R--1-->
!
```

```
!  Modified:
!
!    13 April 2005
!
!  Author:
!
!    John Burkardt
!
!  Parameters:
!
!    Input, real ( kind = 8 ) XN(3), YN(3), the X and Y coordinates
!    of the vertices.  The vertices are assumed to be the images of
!    (0,0), (1,0) and (0,1) respectively.
!
!    Input, real ( kind = 8 ) XSI, ETA, the coordinates of a point
!    in the reference space (and not necessarily within the reference
!    triangle).
!
!    Output, real ( kind = 8 ) X, Y, the coordinates of the corresponding
!    point in the image space.
!
  implicit none

  real ( kind = 8 ) eta
  real ( kind = 8 ) x
  real ( kind = 8 ) xn(3)
  real ( kind = 8 ) xsi
  real ( kind = 8 ) y
  real ( kind = 8 ) yn(3)

  x = ( 1.0D+00 - xsi - eta ) * xn(1) &
                + xsi         * xn(2) &
                      + eta   * xn(3)

  y = ( 1.0D+00 - xsi - eta ) * yn(1) &
                + xsi         * yn(2) &
                      + eta   * yn(3)

  return
end
```

11 Extroduction

This discussion has concentrated on questions of geometry and approximation involved in a finite element calculation. Given a somewhat arbitrary computational region, we created a model described by nodes, with sets of nodes organized into (triangular) elements. The elements were then used to define pieces of a set of linear and quadratic basis functions.

Of course, we intend to use these basis functions to represent pressure and velocity fields, and that will come shortly. But we have so far achieved an important goal. We have demonstrated how a question about arbitrary functions $p(x, y)$ and $\mathbf{v}(x, y)$ over the region Ω can be replaced by a search for suitable vectors of

coefficients *pval* and **vval** which define discretely determined approximating functions $p^h(x,y)$ and $\mathbf{v}^h(x,y)$.

In the next discussion, we will explore how the original classical form of the Navier Stokes equations, which was a statement about the functions $p(x,y)$ and $\mathbf{v}(x,y)$, can be reformulated and reinterpreted as statements about discretized functions $p^h(x,y)$ and $\mathbf{v}^h(x,y)$, which are only approximations to p and v, but which have the advantage of being computable. Thus, we now turn to the issue of determining these coefficient vectors and the discretized approximating functions they define.