Finite Element Treatment of the Navier Stokes Equations, Part III, Solving the discretized Stokes equations

John Burkardt
https://people.sc.fsu.edu/~jburkardt/fem_2005/fem_ns3.pdf

15 August 2021

1 Introduction

At this point in our discussion, we have developed a system of partial differential equations of interest, and we have set up the finite element "machinery" of spatial geometry and basis functions. The next step would be to apply the finite element method to these partial differential equations. But the Navier Stokes equations are nonlinear; this means there will be some extra complications when we try to solve them. Therefore, it seems best to turn first to a related, but linear system, reserving the full Navier Stokes equations for our *second* example of a finite element implementation.

For our first example, we will look at the Stokes equations, which can be thought of as the Navier Stokes equations with the nonlinear term dropped. Since the nonlinear term is multiplied by the Reynolds number, you can assume that the Stokes equations may be suitable for very low Reynolds numbers; more vividly, flows involving low speeds and high viscosity, such as syrup or oil,

There are other benefits of starting with the Stokes equations. Since the Navier Stokes equations are nonlinear, there will be an iteration involved in solving them. The Stokes solution can be used as a reasonable starting value for this iteration. Moreover, the linear system $A\,x = b$ associated with the Stokes equations is very strongly related to the Newton system $F'\,dx = -F$ to be set up for the Navier Stokes equations.

2 The Stokes Equations and a Strong Solution

By dropping the nonlinear momentum term from the Navier Stokes equations, we have the (steady incompressible) Stokes equations:

$$\begin{aligned} -\Delta \mathbf{v} + \nabla p &= 0 \\ \nabla \mathbf{v} &= 0 \end{aligned} \tag{1}$$

Now suppose that we have an arbitrary pair of functions $(\mathbf{v}, p)$. We say this pair of functions is a *strong solution* of the Stokes equations if

$$\begin{aligned} -\Delta \mathbf{v}(x,y) + \nabla p(x,y) &= 0 \\ \nabla \mathbf{v}(x,y) &= 0 \quad \forall (x,y) \in \Omega \end{aligned}$$

Yes, a strong solution is really what we would usually just call a solution; but we define a strong solution first, so that we can now look at ways of defining a weaker solution.

There are some natural reasons for wanting to weaken the definition of a solution. First, a pair of functions cannot be a strong solution to the Navier Stokes equations if its definition fails even at a single point. But we are quite familiar with functions, such as the Dirac delta function, whose pointwise definition is ambiguous, but which have other useful properties, particularly under the integral sign.

Secondly, a pair of functions cannot be a strong solution if it is not suitably continuosly differentiable at every point in the region. But in other areas of mathematics, we are comfortable with functions such as the step function or the absolute value function for which there are pointwise problems. Again, one way to make minor problems of differentiability disappear is to use the smoothing properties of the integral.

Finally, the equations require second order differentiability of $\mathbf{v}$. Rather than merely allowing pointwise or integrable singularities in the second derivatives of $\mathbf{v}$, we would like to be able to lower the equation to first order. This would allow us to admit more interesting kinds of shock waves as solutions.

Thus, before we even begin to try to solve it, we will spend some time considering how we can weaken the concept of a solution to our system.

3 The Weak Form of the Stokes Equations

If a pair of functions is a strong solution to the Stokes equations, then it satisfies those equations at every point in the region. Certainly, this pair will still be a solution if we multiply the Stokes equations by any

function we like. Let us multiply the momentum equation by a function $\psi_i(x,y)$ and the continuity equation a function $\phi_i(x,y)$. Although it might make sense to call these "multiplier functions", we will call them *test functions*. For the moment, we will allow these test functions to be arbitrary, although their names should suggest what we eventually intend them to be! For arbitrary test functions, then, a strong solution will surely satisfy:

$$\begin{aligned} (-\Delta \mathbf{v}(x,y) + \nabla p(x,y))\ \psi_i(x,y) &= 0 \\ \nabla \mathbf{v}(x,y)\ \phi_i(x,y) &= 0 \\ \forall (x,y) \in \Omega & \end{aligned}$$

(To cut down on the length of the equations, we are now going to suppress the explicit dependence of quantities on x and y.)

Since these equations are true at every point in the domain, we can integrate them over the domain, and, it must be still be the case that, for any test functions for which the product remains integrable:

$$\begin{aligned} \int_\Omega (-\Delta \mathbf{v} + \nabla p)\ \psi_i\ d\Omega &= 0 \\ \int_\Omega \nabla \mathbf{v}\ \phi_i\ d\Omega &= 0 \end{aligned}$$

Now it is true that a strong solution of the Stokes equations will satisfy these integrated equations. On the other hand, it should be clear that a pair of functions that satisfy these integrated equations, for all test functions ψ_i and ϕ_i in a "reasonably large" space, such as the space of all polynomials, must be very much like a strong solution. On the other hand, with this integral form of the equations, the solution functions have much more freedom to "misbehave"; that is, there may be points where the functions are undefined, or tend to infinity, and the derivatives may fail to exist, as long as these problems don't affect integrability.

We promised to lower the order of the equation as well, and we now proceed to this last step in weakening the requirements of a solution.

4 Lowering the Order using Integration by Parts

We are going to apply integration by parts to modify our integrated momentum equation a little further. The quantity of concern is the Laplacian operator. To see what we can do, consider the following fact about any pair of smooth (sufficiently continously differentiable) functions $u(x)$ and $v(x)$:

$$\frac{d}{dx}\left(\frac{du}{dx}\ v\right) = \frac{d^2u}{dx^2}v + \frac{du}{dx}\frac{dv}{dx}$$

If we integrate over some interval Ω and rearrange, we have

$$\int_\Omega -\frac{d^2u}{dx^2}v\ dx = -\left(\frac{du}{dx}\ v\right)|_{\partial\Omega} + \int_\Omega \frac{du}{dx}\frac{dv}{dx}\ dx$$

The analogy for our two dimensional region Ω allows us to rewrite the horizontal momentum equation term involving the Laplacian:

$$\int_\Omega -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\ \psi_i d\Omega = -\int_{\partial\Omega} \frac{\partial u}{\partial n}\ \psi_i\ ds + \int_\Omega \nabla u \cdot \nabla \psi_i\ d\Omega$$

Let us consider two assumptions. First, we suppose that the test function ψ_i is not entirely arbitrary, but is, in fact a finite element basis function associated with some node n_i. For our work, this will *always* be the case. Second, suppose that the node n_i is *not* a node on the boundary $\partial\Omega$. (We will look later at how

to handle cases where this is not so.) In that case, the basis function ψ_i will be identically zero along the boundary. Under these two assumptions, the boundary term disappears, and we have:

$$\int_\Omega - \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \psi_i d\Omega = \int_\Omega \nabla u \cdot \nabla \psi_i \, d\Omega$$

We can repeat this process for the vertical momentum equation and combine the two scalar equations back into a vector equation. The vector equation tells us how we may rewrite integrals involving the Laplacian of the velocity field $\mathbf{v}$ and a function ψ_i which vanishes on the boundary:

$$\int_\Omega -\Delta \mathbf{v} \, \psi_i d\Omega = \int_\Omega \nabla \mathbf{v} \, \cdot \nabla \psi_i \, d\Omega$$

This result allows us to write down our final form for the **weak Stokes equations**.

$$\begin{aligned} \int_\Omega \nabla \mathbf{v} \cdot \nabla \psi_i + \nabla p \, \psi_i \, d\Omega &= 0 \\ \int_\Omega \nabla \mathbf{v} \, \phi_i \, d\Omega &= 0 \end{aligned} \tag{2}$$

We say that a pair of functions $(\mathbf{v}, p)$ is a **weak solution** of the Stokes equations if they satisfy the weak Stokes equations for all test functions ψ_i and ϕ_i from a given set, with the functions ψ_i vanishing on the boundary of Ω.

This form of the system is a considerable accomplishment. The smoothing function of the integral allows us to be more daring in what sort of solutions we can consider. It has also allowed us to lower the order of differentiation on our velocity field $\mathbf{v}$, requiring only that $\mathbf{v}$ be a $C^1(\Omega)$ function. We have done this at the cost of assuming that the velocity test functions are differentiable and vanish at the boundary, but in fact, they are polynomials, and they will vanish at the boundary, so that's not a problem at all.

This, then, is our weak form of the Stokes equations. We still haven't quite said what test functions ψ_i and ϕ_i we are going to use. We haven't explained any relationship between these test functions and the solution pair $(\mathbf{v}, p)$. And we haven't really worked out any way yet to actually determine a solution pair. All we know so far is how to judge whether a pair of functions is a strong or weak solution pair, as long as someone is kind enough to tell us what test functions to use.

5 The Discretized Weak Stokes Equations

Everything that we said in the earlier discussion about building suitable finite element spaces for the Navier Stokes variables $\mathbf{v}$ and p also applies to the Stokes equations. Therefore, we will suppose as before that we triangulated the flow region, and constructed the corresponding sets of basis functions $\psi_i(x, y)$ and $\phi_i(x, y)$.

Now we are prepared to relate the Stokes equations and our finite element spaces. We have already become familiar with the idea that we shall look for weak solutions to the equations.

Now we take two further steps, both involving the finite element basis functions.

First, we take the fairly obvious step of specifying the test functions. The test functions for the momentum equations will be the set of velocity basis functions ψ_i; the test functions for the continuity equation will be the pressure basis functions ϕ_i.

Second, we will no longer consider solution functions from some arbitrary space. Instead, the candidate solution functions $(\mathbf{v}, p)$ must be constructed as sums of the finite element basis functions.

In other words, we will only be considering solutions of the weak Stokes equations which can be represented in the form:

$$\mathbf{v}^h(x, y) = \sum_{j=1}^{N_u} \mathbf{vval}_j \, \psi_j(x, y)$$

$$p^h(x,y) \quad = \quad \sum_{j=1}^{N_p} pval_j \; \phi_j(x,y)$$

The superscript is intended to remind us that we have discretized our system, replacing our previous solution space by one in which every element can be described by a finite set of coefficients. In fact, the representation of the solution pair has $2N_u + N_p$ unknown coefficients.

We now have the **discretized weak Stokes equations**, for which we are given sets ψ_i and ϕ_i which will play the roles of both basis and test functions; we wish to find a discrete weak solution $(\mathbf{v}^h, p^h)$ so that, for all the test functions:

$$\begin{aligned} \int_\Omega \nabla \mathbf{v}^h \cdot \nabla \psi_i + \nabla p^h \; \psi_i \; d\Omega &= 0 \\ \int_\Omega \nabla \mathbf{v}^h \; \phi_i \; d\Omega &= 0 \end{aligned} \tag{3}$$

When we form our set of weak Stokes equations by multiplying the (vector) momentum equation by each of the N_u velocity basis functions ψ_i, and the continuity equation by each of the N_p pressure basis functions phi_i, we arrive at a total of $2Nu + N_p$ equations.

Thus, the number of unknown coefficients is matched by the number of equations, and this has a comforting feeling of completeness to it; we know that we probably have just enough information to determine the unknown coefficients. We have transformed a set of partial differential equations with an infinite number of degrees of freedom into a relatively puny set of algebraic equations, which, under reasonable assumptions, can be guaranteed to be solvable.

Of course, anybody can replace a hard problem by an easy, solvable one. But the test is, does solving the easier problem tell you anything about the hard problem you gave up on? At this point, I will simply promise you that, as long as we are willing to solve the finite element version of the Stokes equations on a "fine enough" grid, the errors in our approximate solution will become arbitrarily small.

The process of multiplying the equations you are trying to satisfy by a set of test functions, and then integrating, is called the *Galerkin method*. What the method is actually doing is requiring that the error be orthogonal to the space spanned by the test functions. Here, our "dot product" is simply the L2 inner product

$$< u^h, v^h > \equiv \int_\Omega u^h(x,y) \; v^h(x,y) \; d\Omega$$

When, as in our case, the test functions are also the set of basis functions used to represent solutions, the process is called the *Petrov-Galerkin method*.

6 The Form of One Discretized Weak Continuity Equation

Let's look closely at the form of the discretized weak continuity equation associated with a given pressure node n_i. This pressure node has an associated pressure basis function $\phi_i(x,y)$, and therefore our system of finite element equations will include a copy of the continuity equation, multiplied by $\phi_i(x,y)$ and integrated over the flow region:

$$\int_\Omega \nabla \mathbf{v}^h \; \phi_i \; d\Omega = 0$$

In terms of the horizontal and vertical velocity component functions $u(x,y)$ and $v(x,y)$, this equation has the form:

$$\int_\Omega \left(\frac{\partial u^h}{\partial x} + \frac{\partial v^h}{\partial y} \right) \; \phi_i \; d\Omega \quad = \quad 0$$

and if we replace each component function by its underlying finite element form, we have:

$$\int_\Omega \sum_{j=1}^{N_p} \left(uval_j \, \frac{\partial \psi_j}{\partial x} + vval_j \, \frac{\partial \psi_j}{\partial y} \right) \, \phi_i \, d\Omega \quad = \quad 0$$

This system can be regarded as a linear equation in the unknown velocity coefficients. In particular, the coefficient A_{ij} of $uval_j$ is

$$\int_\Omega \frac{\partial \psi_j}{\partial x} \, \phi_i \, d\Omega$$

and the coefficient A_{ij} of $vval_j$ is

$$\int_\Omega \frac{\partial \psi_j}{\partial y} \, \phi_i \, d\Omega$$

But we have formulas for the basis functions, and we could even work out a formula for the partial derivatives, so if we could carry out the integration, we'd be able to determine these numbers, and hence we could reduce this weak continuity equation to a linear equation.

7 Sparseness

Although the continuity equation seems to involve all the velocity basis functions, we should note that *most of the coefficients will actually be zero*. This is fairly easy to see. The weak continuity equation associated with node n_i involves the integral of the product of the pressure basis function ϕ_i and derivatives of the velocity basis function ψ_j. We know that ϕ_i is only nonzero in elements which include node n_i as a vertex.

So consider the coefficient of an arbitrary velocity basis function ψ_j, associated with velocity node n_j. If nodes n_i and n_j are not both nodes in some common element, then in fact, in every element, either ϕ_i or ψ_j is identically zero. And of course, if ψ_j is uniformly zero in an element, then so are its partial derivatives. So if a velocity node is "too far" from the pressure node, then its basis function does not participate in the continuity equation.

Thus, the continuity equation associated with node n_i will actually only have a few "interesting" (that is, nonzero) coefficients. Suppose, for instance, that we begin with grid of squares, and split each square diagonally into two triangular elements. Then, except near the boundary, a pressure node will be a member of 6 elements. These elements involve a total of 19 velocity nodes. Thus, the continuity equation associated with this pressure node will have (at most) 38 nonzero coefficients, associated with the adjacent velocity nodes. This fact will remain true no matter how many elements are involved in the total mesh.

Althought they are more complicated, you may also verify that each momentum equation also has a similar property. Each momentum equation only has a limited number of nonzero coefficients. These nonzero coefficients are associated with velocity and pressure basis functions whose defining nodes are "nearby" the node associated with the test function for the given momentum equation.

This fact means that the linear system defining the finite element coefficients is *sparse*, or mostly zero. It is generally true that a sparse linear system can be constructed, stored, factored and solved much more efficiently than a general linear system. This counts as another advantage of the finite element method.

Even if you don't want to take full advantage of sparseness, the linear system is also most likely *banded*; that is, there are some lower and upper bandwidths **ml** and **mu**, so that any entry is guaranteed to be zero if you it is more than **ml** entries below the diagonal or **mu** entries above it. The exact values of the bandwidths usually need to be figured out after you've set up your geometry. In a 1D problem, the matrix really is tightly banded and the matrix can be stored in a very compact form. In a 2D problem, the band structure is less compact; there are bands near the diagonal, but also several more sets at some distance, with many zeros in between. Instead of a simple band matrix approach, a more sophisticated technique can be used to store the nonzero information. In 3D, trying to use a band structure for the matrix will probably not save enough time or space to be worthwhile, and instead a sparse structure is really necessary!

Once we've stored the linear system in some fashion, we have to come up with a solution technique. Depending on our storage format and solution method, we may need to allocate yet more space for "fill in" values, which occur, for example, in Gaussian elimination.

8 The Form of One Discretized Weak Momentum Equation

Writing out a sample momentum equation, we have

$$\int_\Omega \nabla \left(\sum_{j=1}^{N_u} \mathbf{vval}_j \ \psi_j \right) \cdot \nabla \psi_i + \nabla (\sum_{i=1}^{N_p} pval_j \ \phi_j) \ \psi_i \ d\Omega = 0$$

We can interchange summation and differentiation:

$$\int_\Omega \left(\sum_{j=1}^{N_u} \mathbf{vval}_j \ \nabla \psi_j \right) \cdot \nabla \psi_i + (\sum_{i=1}^{N_p} pval_j \ \nabla \phi_j) \ \psi_i \ d\Omega = 0$$

Again, this is a linear system for the finite element coefficients. The multiplier A_{ij} of a horizontal velocity coefficient $uval_j$ or a vertical velocity coefficient $vval_j$ is

$$\int_\Omega \nabla \psi_j \cdot \nabla \psi_i \ d\Omega$$

The multiplier A_{ij} of a pressure coefficient $pval_j$ is

$$\int_\Omega \nabla \phi_j \ \psi_i \ d\Omega$$

9 Numerical Quadrature

The discretized Stokes equations have been transformed into a linear system for the coefficients of the basis functions. We know formally what these coefficients are, but we need a way to evaluate the integrals so that we can do numerical work.

The integrals we deal with are defined over the entire region Ω, but in fact, are generally only nonzero in a much smaller region. That region, in turn, is simply the set of triangular elements over which the appropriate test function is nonzero. By the linearity of the integral, we really only need to be able to figure out how to integrate a general function over a general triangular element. Since we wish to be able to deal with very general problems, we will not expect to be able to evaluate integrals exactly. Instead, we will be satisfied with an approach that allows us to approximate the integral of almost any integrand function, in such a way that the approximation error is small, or can be made smaller if we are unsatisfied with the initial approximation.

A straightforward approach begins with a numerical quadrature rule for the unit triangle Δ defined by

$$\Delta = \{ (\xi, \eta) : 0 <= \xi, \ 0 <= \eta, \ \xi + \eta <= 1 \}$$

A quadrature rule of order nq specifies a set of abscissas (ξ_i, η_i) in the unit triangle, and weights μ_i so that the integral of a function $f(x, y)$ over the unit triangle can be approximated by

$$\int_\Delta f(\xi, \eta) \, d\xi d\eta \approx \sum_{i=1}^{nq} \mu_i \ f(\xi_i, \eta_i)$$

For instance, a simple quadrature rule of order $nq = 3$ uses the three vertices as abscissas, with equal weights of $\frac{1}{6}$. This "vertex quadrature rule" is actually a fairly poor rule. It's no better, in fact, than

using the one point centroid rule. And if we are willing to use $nq = 3$ points in a quadrature rule, the rule that uses the midpoints of the sides as quadrature points is of higher accuracy than the vertex quadrature rule. In general, a higher order rule, such as with $nq = 7$ or $nq = 13$ is preferred to improve the degree of approximation.

To approximate integrals over a general triangular element T, with vertices (x_1, y_1), (x_2, y_2), and (x_3, y_3), we need to adjust the quadrature rule. Each of the unit triangle abscissas (ξ, η) is transformed to an abscissa (x, y) in the general element as follows:

$$\begin{aligned} x &= \xi x_1 + \eta x_2 + (1 - \xi - \eta) x_3 \\ y &= \xi y_1 + \eta y_2 + (1 - \xi - \eta) y_3 \end{aligned}$$

Each corresponding weight μ for the unit triangle is transformed to a weight w for the general element by

$$w = \mu \; (\; x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \;)$$

which simply multiplies the old weight by the ratio of the areas of T and Δ.

For a point (x, y) in a given triangle T, the quantities $(\xi, \eta, 1 - \xi - \eta)$ defined above are known as the *barycentric coordinates* of the point with respect to the triangle T. Imagine connecting (x, y) to the three vertices of T, forming three triangles. The barycentric coordinates are the relative areas of each of the resulting subtriangles, compared to the area of T. The idea of barycentric coordinates appears in many contexts within the finite element method, as well as in the theory of interpolation.

10 Assembly Code

Here is sample MATLAB code to assemble the finite element equations for the Stokes problem. I have removed most of the comments, but you should be able to figure out what is going on by making correspondences with the formulas we are trying to implement. Note that **qbf** and **lbf** are functions that evaluate the quadratic (velocity) and linear (pressure) basis functions at a given point. We also include a few lines to take care of source terms, which are discussed in a following section. Finally, note that while every node has an associated velocity, only some nodes have associated pressure; as we loop through test and basis functions, we must always ask ourselves, is this a node with 2 or with 3 degrees of freedom? The **indx** array is used to keep track of the degrees of freedom; a positive entry in this array indicates the index of the equation used to constrain the corresponding degree of freedom.

```
function [ a, f ] = assemble ( node_num, node_xy, nnodes, element_num, ...
  element_node, nq, wq, xq, yq, element_area, indx, nunk )

  f(1:nunk) = 0.0;
  a(1:nunk,1:nunk) = 0.0;

  for element = 1 : element_num

    for quad = 1 : nq

      x = xq(quad,element);
      y = yq(quad,element);
      w = element_area(element) * wq(quad);

      for test = 1 : nnodes

        in = element_node(test,element);
```

```
        iu = indx(1,in);
        iv = indx(2,in);
        ip = indx(3,in)

        [ bi, dbidx, dbidy ] = qbf ( x, y, element, test, node_xy, ...
          element_node, element_num, nnodes, node_num );

        if ( 0 < ip )
          [ qi, dqidx, dqidy ] = lbf ( x, y, element, test, node_xy, ...
            element_node, element_num, nnodes, node_num );
        end
%
%  Handle source terms.
%
        [ u_rhs, v_rhs, p_rhs ] = rhs ( x, y );

        f(iu) = f(iu) + w * u_rhs * bi;
        f(iv) = f(iv) + w * v_rhs * bi;
        if ( 0 < ip )
          f(ip) = f(ip) + w * p_rhs * qi;
        end

        for basis = 1 : nnodes

          jn = element_node(basis,element);
          ju = indx(1,jn);
          jv = indx(2,jn);
          jp = indx(3,jn);

          [ bj, dbjdx, dbjdy ] = qbf ( x, y, element, basis, node_xy, ...
            element_node, element_num, nnodes, node_num );

          if ( 0 < jp )
            [ qj, dqjdx, dqjdy ] = lbf ( x, y, element, basis, node_xy, ...
              element_node, element_num, nnodes, node_num );
          end

          a(iu,ju) = a(iu,ju) + w * ( dbidx * dbjdx + dbidy * dbjdy );
          if ( 0 < jp )
            a(iu,jp) = a(iu,jp) + w * bi * dqjdx;
          end

          a(iv,jv) = a(iv,jv) + w * ( dbidx * dbjdx + dbidy * dbjdy );
          if ( 0 < jp )
            a(iv,jp) = a(iv,jp) + w * bi * dqjdy;
          end

          if ( 0 < ip )
            a(ip,ju) = a(ip,ju) + w * qi * dbjdx;
            a(ip,jv) = a(ip,jv) + w * qi * dbjdy;
          end
```

```
        end
      end
    end
  end
```

11 Source Terms

We have seen how the discretized Stokes equations may be regarded as a linear system for the finite element coefficients C. We might write this linear system as

$$AC = F$$

We have seen how the entries of A come from terms in the momentum and continuity equations. But what about the right hand side F? At the moment, this right hand side is zero, which means our coefficients C must be zero as well. It seems like we are still missing some key ingredient of our formulation.

The first thing to note is that we have looked at the homogeneous Stokes equations. That is, the original partial differential equations had zero right hand sides, and this fact has carried over to the discretized system. It would, in fact, be possible to supply nonzero right hand side functions $f_1(x, y)$ and $f_2(x, y)$.

These source terms might come from extra physical effects to be modeled. They might also come simply by deciding that you want the solution $(\mathbf{v}, p)$ to have a particular form. If you simply plug your desired solution into the homogeneous equations, it will almost surely not satisfy the equations, producing instead two residuals. But you can now simply replace the zero right hand sides of the equations by functions $f_1(x, y)$ and $f_2(x, y)$ which are equal to these two residuals. Your desired solution will now be consistent with the new, nonhomogeneous system. (This is one way to manufacture a test case for your code.)

However, if you specify source terms in your continuous pde's, then this will, of course, affect the discretized pde's in the obvious way. In particular, the nonhomogeneous discretized Stokes equations will be:

$$\begin{aligned} \int_\Omega \nabla \mathbf{v}^h \cdot \nabla \psi_i + \nabla p^h \psi_i \, d\Omega &= \int_\Omega f_1(x,y) \psi_i d\Omega \\ \int_\Omega \nabla \mathbf{v}^h \phi_i \, d\Omega &= \int_\Omega f_2(x,y) \phi_i d\Omega \end{aligned} \tag{4}$$

It is easy to evaluate the source term integrals that define the nonzero entries F_i of the linear system, in the same way that you work out the integrals that define the coefficient matrix entries $A_{i,j}$. And if the right hand side of your linear system is nonzero, then at least some of the finite element coefficients will be nonzero, so our solution will finally be more interesting.

12 Boundary Conditions

Perhaps a more natural way in which we can avoid solving a homogeneous problem is to specify boundary conditions, that is, values that the solution must attain at certain points of the boundary of the region. In Stokes flow, it is common to specify the velocity, or its tangential or normal component, at every point on the boundary.

If part of the boundary is a physical wall, a typical condition is that the flow is exactly zero along the boundary. This is sometimes called a *no-slip* condition. In some cases, a wall may be treated as impenetrable, but "slippery"; in that case, we specify that the velocity component normal to the wall must be zero, while

the tangent component is not specified. This "free slip" condition might also be applied at a free surface of the liquid, or a plane of symmetry.

It is frequently the case that, along some portion of the boundary, we are pumping fluid into the region. This is known as an *inflow* boundary condition, and we are presumably able to specify the value of both components of the velocity vector at any point along this inflow interface.

Finally, if we have inflow, we most likely also have outflow, and hence we generally have to say something about the behavior of the outflow. It's natural to assume that we can specify the inflow, but it is not so reasonable to say the same thing about the outflow. You can squirt water into a box at any angle and speed you like; but what comes out (eventually) will usually be a surprise! The physics of outflow boundary conditions are shaky; therefore, the outflow conditions we impose are usually much weaker; when possible, we try to impose the outflow boundary conditions far enough downstream, so that we don't worry about nonphysical effects.

A common choice for an outflow boundary condition might specify something like "straight flow", which says that along the outflow interface, the normal derivative of the velocity (relative to the interface) is zero, and the tangential velocity is zero. Straight flow doesn't guarantee outflow; it's quite possible to have "negative outflow" at an interface. This is usually undesirable, since it means that your problem is pulling in phantom liquid. All you can do is watch for such cases, and adjust your boundary conditions, initial conditions, or other quantities to avoid such a problem.

For the pressure, it is common to specify a single value at one point, or to specify some normalizing condition, such as requiring that the integral of the pressure over the flow region is zero.

Since we are desperately trying to keep things simple, let us assume for the moment that the only boundary conditions we want to apply are *Dirichlet conditions*, that is, constraints that specify the value of a solution component, but that do not involve derivatives of the solution.

Normally, our Dirichlet boundary conditions will be specified in continuous form. For instance, we might specify that, along some vertical portion of the boundary, the horizontal velocity u is equal to $y(1-y)$ and the vertical velocity component is 0. We need to discretize our boundary conditions in the same way as we discretized our pde's. In this case, it is natural to interpret the boundary conditions as saying that, for each node n_i along the given portion of the boundary, the finite element solution should agree with the boundary function. Thus, in particular, we would want $u_i = y_i(1-y_i)$ and $v_i = 0$.

Our discretization of the boundary condition amounts to replacing the continuous boundary function $y(1-y)$ by the piecewise quadratic interpolant generated by our finite element geometry along the boundary nodes. For this particular boundary function, there is no loss of information because we can capture a continuous quadratic function exactly. But if our boundary function was, say, $\sin(y(1-y))$, then it should be clear that our discretized boundary condition will not be dealing with the original function, but with a discrete interpolant. Usually, this is not a serious problem, and refining the mesh will reduce the discrepancy between the true and discretized boundary functions.

Now our original finite element system already had exactly one equation for every unknown. Adding boundary conditions increases the number of equations without increasing the number of unknowns. How do we reduce the number of equations, and which ones do we need to get rid of?

In our Dirichlet boundary condition setting, we can break down our boundary conditions into conditions that constrain individual finite element coefficients. To say, for instance, that u has a given value along the left side of the region is to say that for each velocity node along the left side, the corresponding horizontal velocity coefficient $uval_i$ has the appropriate value.

If we suppose we have already set up our finite element equations, then there is, in fact, an equation in our system associated with the value of $uval_i$, namely, the horizontal component of the momentum equation which was multiplied by the velocity basis function ψ_i associated with node n_i. This equation is no longer appropriate; it should be replaced by the information that the boundary condition is supplying. But to do so is easy. We can simply replace the original equation by the following

$$1 \; uval_i = y_i(1-y_i)$$

In other words, we wipe out the original row of A and the corresponding entry of F. Then we put a 1 in the

diagonal position of A, and the boundary value into the entry of F.

Doing this seems complicated, but it is much simpler than trying to apply the boundary condition by eliminating the known value from the linear system. The size and shape of the linear system is preserved.

Similarly, the boundary condition for v_i would result in replacing the original finite element equation by

$$1 \; vval_i = 0$$

A condition on the pressure is handled similarly, except that we will be replacing an instance of the continuity equation that was associated with the test function ϕ_i associated with the pressure degree of freedom that we are constraining.

13 Sample Boundary Condition Code

Here is a sample MATLAB routine which enforces Dirichlet boundary conditions by modifying the finite element matrix and right hand side. It is assumed that the nodes are arranged in a rectangular **2ny-1** by **2nx-1** array, and that **indx** keeps track of the index of the equation associated with the horizontal and vertical velocity and pressure associated with each node.

```
function [ a, f ] = boundary ( nx, ny, node_num, node_xy, indx, a, f )
%
%  Consider each node.
%
  node = 0;

  for row = 1 : 2 * ny - 1
    for col = 1 : 2 * nx - 1
      node = node + 1;
%
%  The value of U is to be set to X^2+Y^2 at every boundary node.
%
      if ( row == 1 | row == 2 * ny - 1 | col == 1 | col == 2 * nx - 1 )
        x = node_xy(1,node);
        y = node_xy(2,node);
        i = indx(1,node);
        a(i,1:neqn) = 0.0;
        a(i,i) = 1.0;
        f(i) = x^2 + y^2;
      end
%
%  The value of V is to be set to 0 at every node on the left and right.
%
      if ( col == 1 | col == 2 * nx - 1 )
        i = indx(2,node);
        a(i,1:neqn) = 0.0;
        a(i,i) = 1.0;
        f(i) = 0.0;
      end

    end
  end
%
```

```
%  The value of P is to be set to 1 at node number 1.
%
  node = 1;
  i = indx(3,node);
  a(i,1:neqn) = 0.0;
  a(i,i) = 1.0;
  f(i) = 1.0;
```

14 Outroduction

The Stokes equations have gotten us a long way - from a pair of partial differential equations, through a weak formulation, and a discretization, to a set of algebraic linear equations. Because the Stokes equations were reasonably simple, we could spend some time worrying about the form of particular elements of the finite element matrix, and the effect of adding source terms and boundary conditions.

We are now ready to turn to the Navier Stokes equations. Everything we have learned about the Stokes equations will be useful in this new setting. The main new feature will, of course, be the addition of the nonlinear term to the momentum equations. This means that we can no longer simply set up a linear system for the finite element coefficients and solve it. Instead, we will encounter a nonlinear system whose solution requires a good starting point, and the application of Newton's method or some other iterative procedure.

At that time, we will also return to the question of boundary conditions, and explain how to deal with Neumann boundary conditions that involve constraints on the derivative of the velocity.