Finite Elements for the Steady Stokes Equations

John Burkardt
Department of Scientific Computing
Florida State University

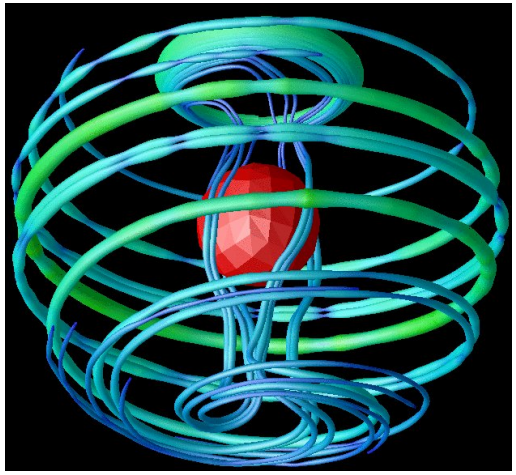
.....

12:30-1:45, 23 June 2014
ISC 5907

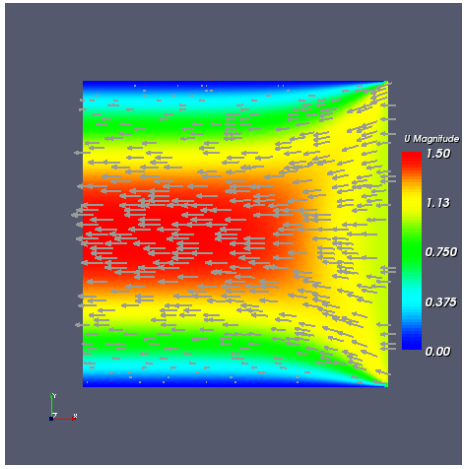
.....

https://people.sc.fsu.edu/~jburkardt/presentations/...
stokes_2014_fsu.pdf

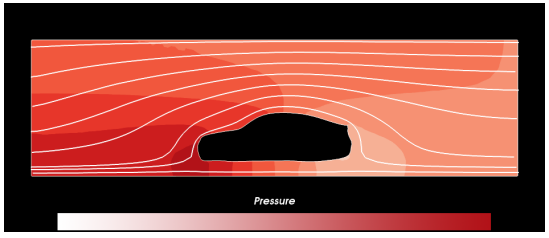
Fluid Flow Problems and Fluid Flow Solvers



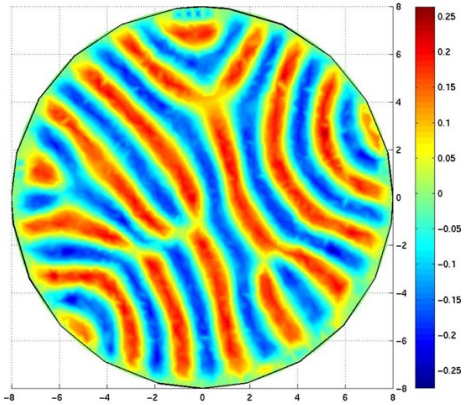
Deal.II



Fenics



FreeFem++



Ifiss

INTRO: Equations of Fluid Motion

The equations that describe fluid flow are more complicated, in a number of ways, than what we have seen for model problems such as the Poisson equation in 1 or 2 dimensions.

Let's take a look at a standard model for fluid flow, the time-dependent Navier-Stokes equations.

We'll simplify this system as much as possible, to the steady Stokes equations, which are still complicated enough to introduce some of the ideas you need in order to apply the finite element method to fluid flow.

- **Equations of Fluid Motion**
- A Finite Element Formulation
- Computing Basis Functions
- Assembling the Matrix

EQUATIONS: The Navier Stokes Equations

Any study of fluid flow starts with the **Navier-Stokes** equations:

$$\rho \mathrm{v}_t - \rho\nu\Delta \mathrm{v} + \rho(\mathrm{v}\cdot\nabla)\mathrm{v} + \nabla p = \mathrm{f} \quad \textit{(momentum equations)}$$
$$\rho_t + \nabla\cdot(\rho \mathrm{v}) = 0 \quad \textit{(continuity equation)}$$

This is a relatively simple version!

We could add compressibility, heat transfer, turbulence.

Notice our hidden friend the (time dependent) Poisson equation:

$$\rho \mathrm{v}_t - \rho\nu\Delta \mathrm{v} = \mathrm{f}$$

- v is the 1D/2D/3D velocity vector: u, or (u,v) or (u,v,w);
- p is the pressure;
- ρ is the (constant?) fluid density;
- ν is the kinematic viscosity (stickiness);
- f represents body forces such as gravity.

EQUATIONS: Unsteady Compressible Navier Stokes

Our first simplification is to work in 2D.

Here are the time-dependent compressible Navier Stokes equations, in 2D Cartesian coordinates, with gravity the only external force.

$$\rho\frac{\partial u}{\partial t} - \rho\nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + \rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} = 0$$

$$\rho\frac{\partial v}{\partial t} - \rho\nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) + \rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = -\rho g$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0$$

EQUATIONS: Simplifications

In many flow problems, there is a transient solution that gradually settles down to a long-term unvarying flow.

The equation for this *steady-state* flow is found by dropping the time derivative from our equation.

If our fluid is essentially constant density (like water, but not like air) we can divide through by the constant ρ.

We can replace $\frac{p}{\rho}$ by p (rescaling the pressure units);

Gravitational force can be absorbed into pressure.

We'll just write f_u, f_v for the right hand sides.

EQUATIONS: Steady, Incompressible Navier-Stokes

Now we have the steady incompressible Navier Stokes equations.

The two momentum equations have the linear diffusion term (multiplied by ν) and nonlinear terms, such as $u\frac{\partial v}{\partial x}$.

$$-\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+\frac{\partial p}{\partial x}=f_u$$

$$-\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+\frac{\partial p}{\partial y}=f_v$$

The continuity equation enforces the incompressibility constraint. Over any closed region, the flow in must equal the flow out.

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0$$

EQUATIONS: The Reynolds Number

The viscosity ν controls the balance between smoothing from diffusion and disruption from nonlinear momentum terms.

If ν decreases, the physical system becomes more irregular, the PDE less stable, and the discretized computer model can fail. Experimentally, a smooth flow becomes turbulent and complex.

The Reynolds number Re is a dimensionless quantity that estimates this irregularity on the same scale for all flows:

$$Re = \frac{||v||L}{\nu}$$

where L is a characteristic length.

Mathematicians tend to solve problems with $Re = 1$; a swimmer in a pool might suggest a value of $Re \approx 1,000,000$. Turbulent flow typically arises at Re values in the thousands or higher.

EQUATIONS: Steady, Incompressible Stokes

For "small" values of the velocity, the linear terms are more important than the nonlinear terms, which are quadratic. By dropping these terms, we arrive at an equation for the linearized flow, known as the steady *Stokes equations*:

$$\begin{aligned} -\nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{\partial p}{\partial x} &= f_u \\ -\nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{\partial p}{\partial y} &= f_v \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

The Stokes equations are also useful when solving the Navier-Stokes problem, because they can give a decent starting point for a Newton iteration.

EQUATIONS: A Problem to Solve

Whatever version of the flow equations we choose, we must formulat a mathematical model of the problem:

- the viscosity ν;
- the density ρ;
- body forces;
- the domain Ω;
- the boundary Γ and the boundary conditions;
- the time interval and initial conditions.

To solve the mathematical model, we make a discrete version.

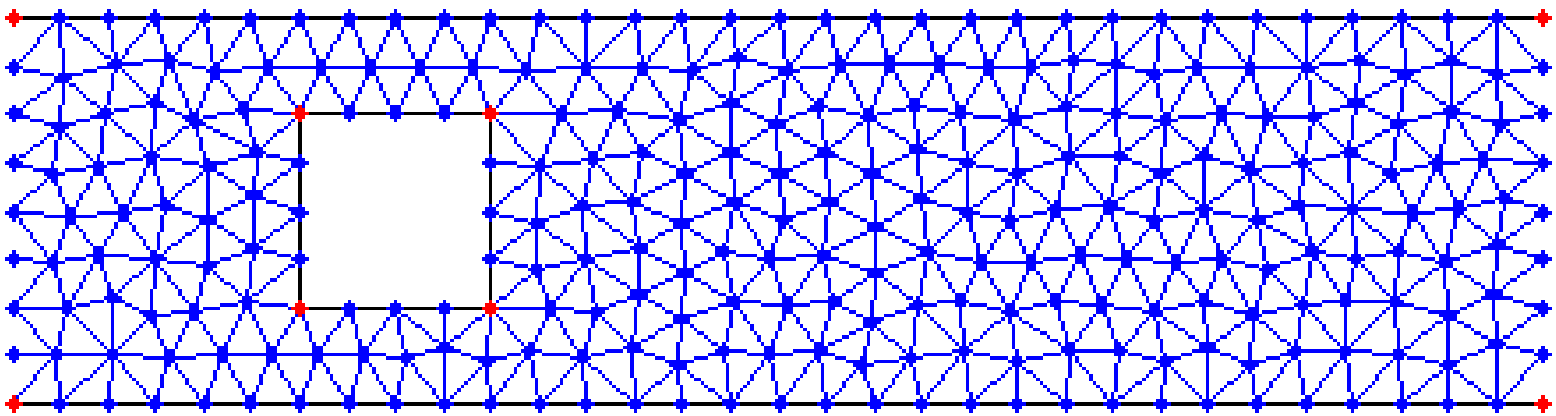
- Equations of Fluid Motion
- **A Finite Element Formulation**
- Computing Basis Functions
- Assembling the Matrix

FEM: A Test Problem

The finite element method begins by discretizing the region.

Here is a rectangular channel with a square obstacle. Top and bottom are walls, flow enters from the left and exits on the right.

I've created a grid using the MATLAB program **mesh2d**:



Typically, the user simply outlines the region, and a meshing program automatically generates the elements of area.

http://www.mathworks.com/matlabcentral/fileexchange/25555-mesh2d-automatic-mesh-generation

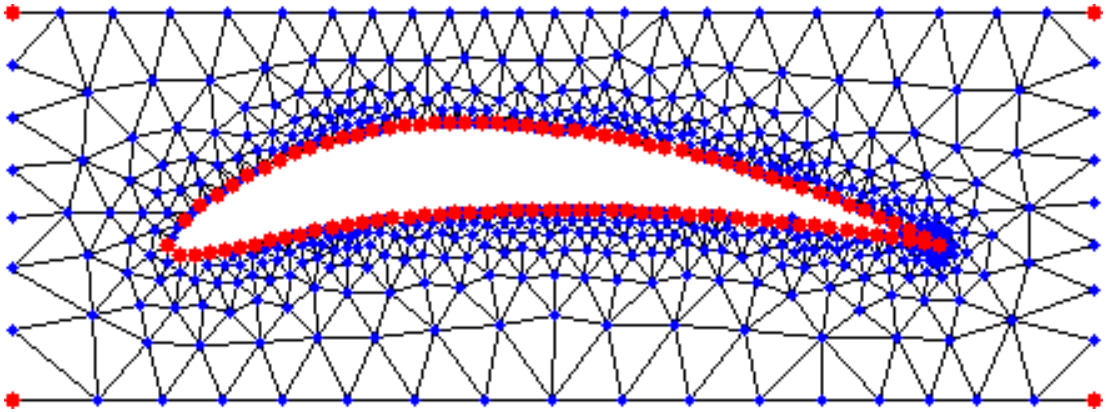
FEM: MESH2D Can Make a Mesh For Us

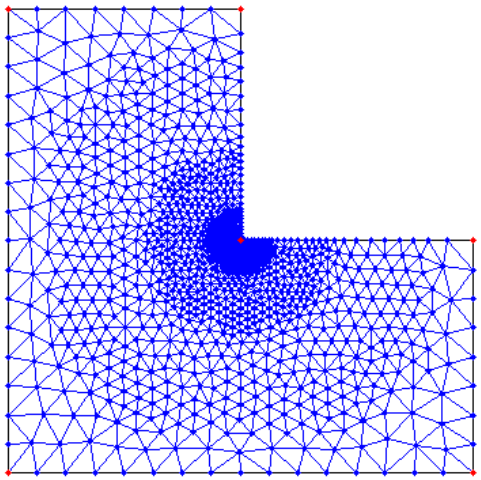
Darren Engwirda's Matlab code **mesh2d** set up this grid:

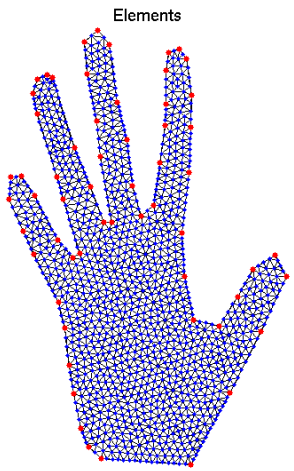
```
v = [ 0.0, -1.0; 8.0, -1.0; 8.0, +1.0; 0.0, +1.0;
      1.5, -0.5; 1.5, +0.5; 2.5, +0.5; 2.5, -0.5 ];
      <-- vertices

e = [ 1, 2;  2, 3;  3, 4;  4, 1;   <-- clockwise
      5, 6;  6, 7;  7, 8;  8, 5 ]; <-- counter-clockwise
      <-- vertex pairs form boundary edges

hdata = [];
hdata.hmax = 0.25;     <-- Maximum element size

[ p, t ] = mesh2d ( v, e, hdata );<--points, triangles.
```

FEM: Interesting Regions

MESH2D can automatically handle more interesting regions:







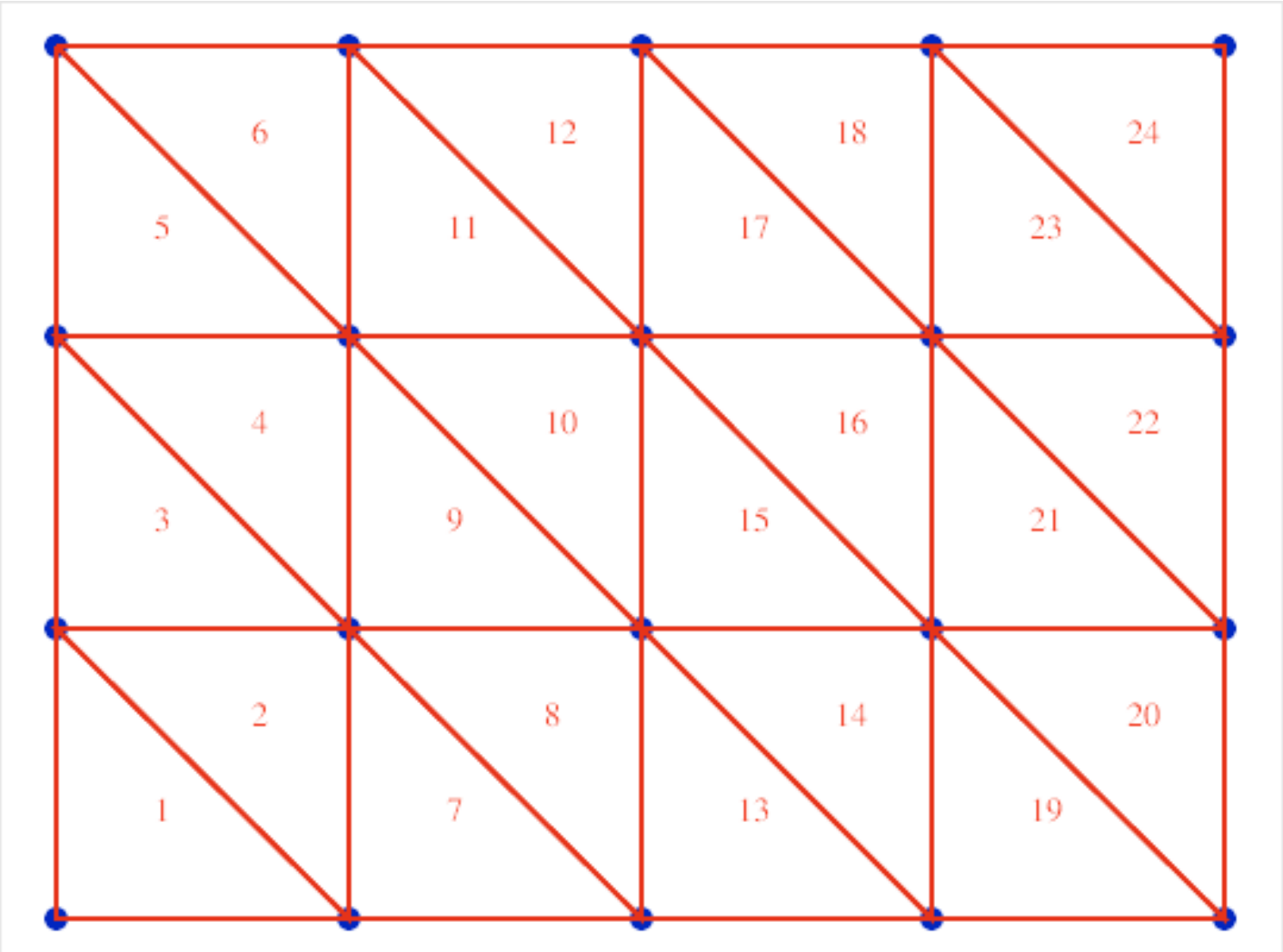
FEM: A Linear Grid of Triangles

Using a program like MESH2D, we can take a description of a region Ω, perhaps outlined by a set of vertices, and produce a set of *nodes* which can be triangulated so that triplets of nodes define triangular *elements*.

A triangulation allows us, in a natural way, to define linear basis functions $\phi_i(x, y)$, which are 1 at node i, 0 at all other nodes, and linear over each element.

For reasons I will explain in a minute, let's call the nodes we've just created *pnodes*. This name is meant to suggest that these nodes are associated with the pressure variable.

FEM: A Pressure Grid



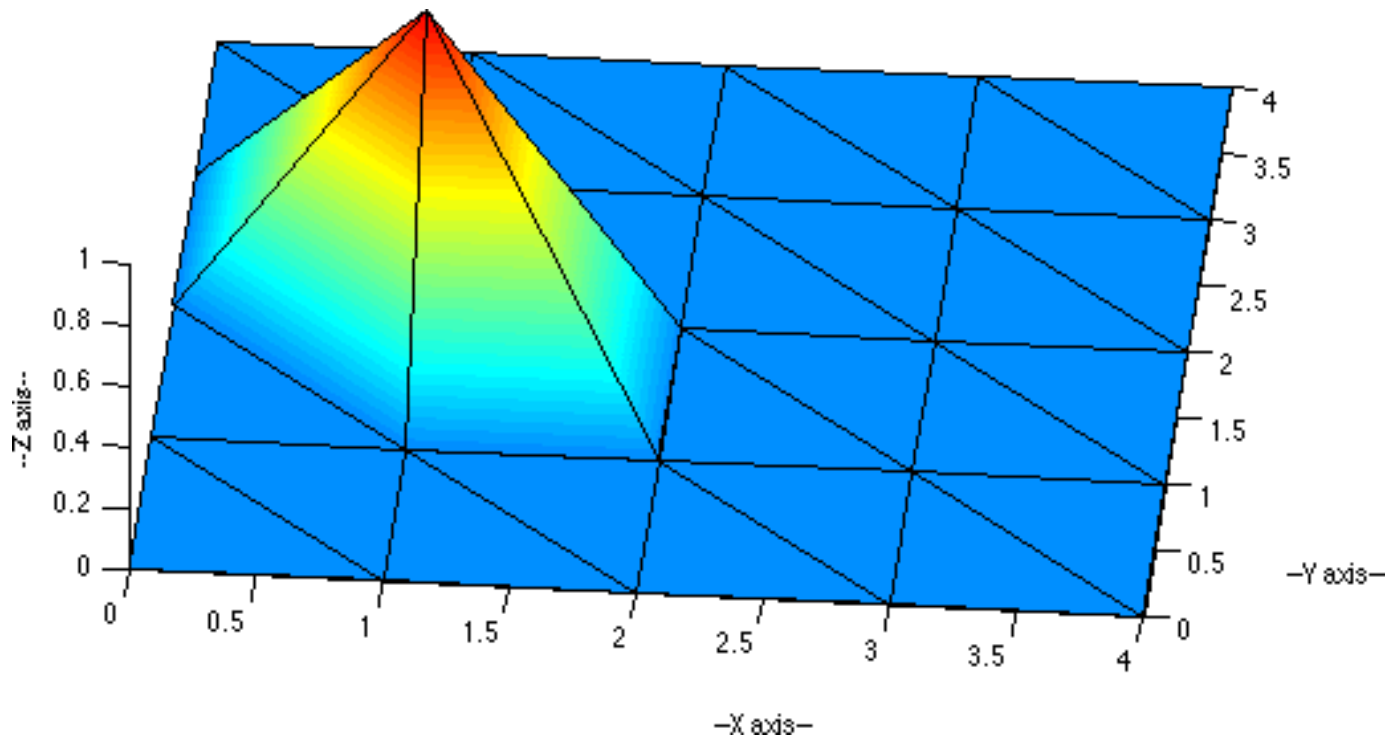
FEM: Pressure Representation

We need to represent our variables as linear combinations of basis functions. The easy case is the pressure p. We can take this to be a linear combination of piecewise linear basis functions $\phi_j(x, y)$,

$$p = \sum_{j=1}^{\text{pnodes}} c_j \, \phi_j(x, y)$$

where the j-th basis function is associated with the j-th pnode.

FEM: A Linear Basis Function



http://people.sc.fsu.edu/~jburkardt/m_src/fem2d_basis_t3_display/fem2d_basis_t3_display.html

FEM: The LBB Condition

For the Navier-Stokes equations, it turns out that you cannot arbitrarily pick the basis functions.

The problem is related to the "“Ladyzhenskaya-Babuska-Brezzi" ("LBB") or "inf-sup" condition. One way to avoid it uses a Taylor-Hood pair of basis functions for the pressure and velocity.

In a typical Taylor-Hood scheme, the polynomial degree of the pressure basis functions is one lower than that used for velocities.

We are using piecewise linear functions for pressure, and so it turns out we should use piecewise quadratic functions for velocity.

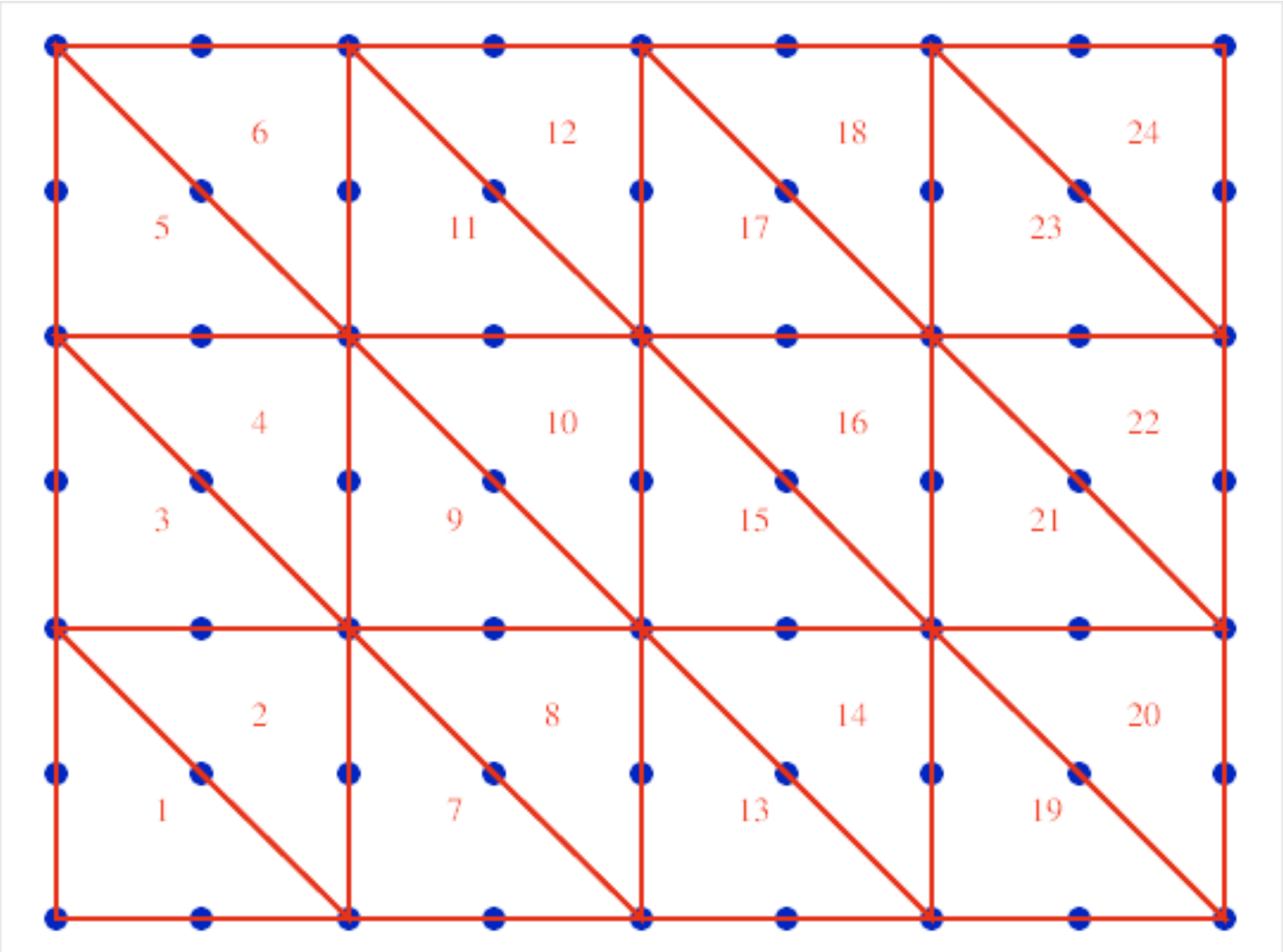
FEM: Velocity Grid

Now we will construct a second grid that is a sort of refinement of the first. The set of nodes in this grid will be called *vnodes*, because they will be associated with velocities. We start by including all the pnodes, but we create a new node at the midpoint of every element edge, and add all these nodes as well.

We can look at this procedure as involving two grids, one for pressure and one for velocities. The two grids are nested in an interesting way.

The velocities will "live" on a grid of six-node triangles. These triangles share their vertices with the three-node pressure triangles. But the six-node triangles can be used to define basis functions $\psi_j(x, y)$ which are 1 at node j, zero at all other nodes, and a quadratic function over each element.

FEM: Velocity Grid



FEM: Velocity Representation

Our velocities will similarly be represented using the quadratic ψ functions. Since velocity is a vector, we can think of it as having components (u, v). Our representation can then be written:

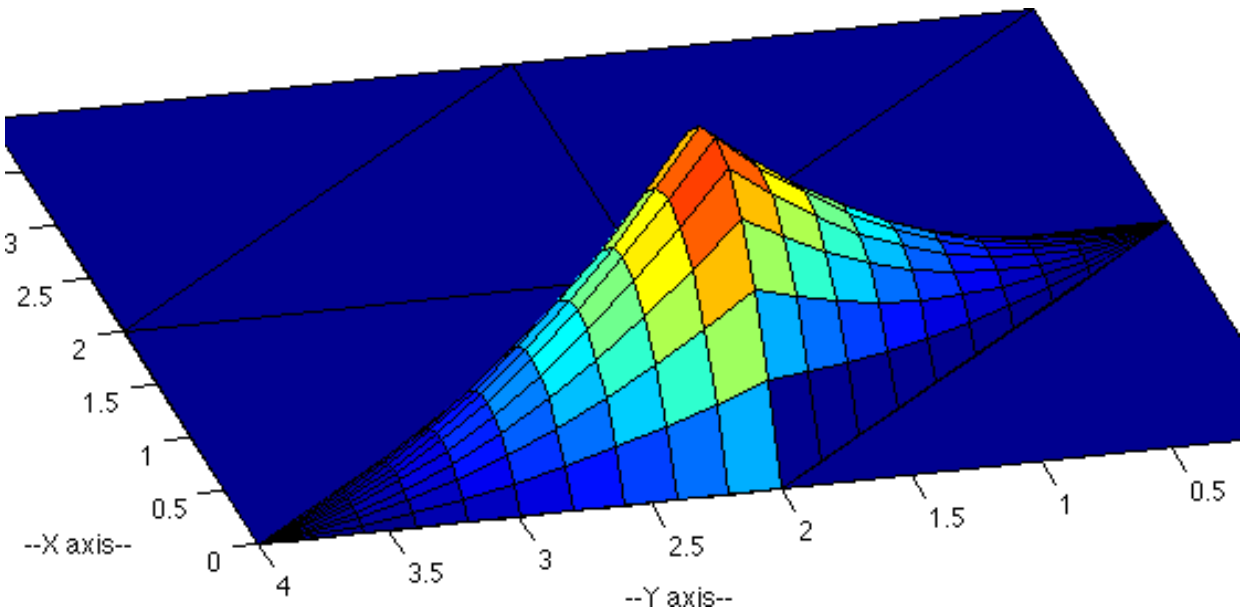
$$u = \sum_{j=1}^{\text{vnodes}} a_j \, \psi_j(x, y)$$

$$v = \sum_{j=1}^{\text{vnodes}} b_j \, \psi_j(x, y)$$

or

$$\begin{pmatrix} u \\ v \end{pmatrix} = \sum_{j=1}^{\text{vnodes}} \begin{pmatrix} a_j \\ b_j \end{pmatrix} \psi_j(x, y)$$

FEM: A Quadratic Basis Function

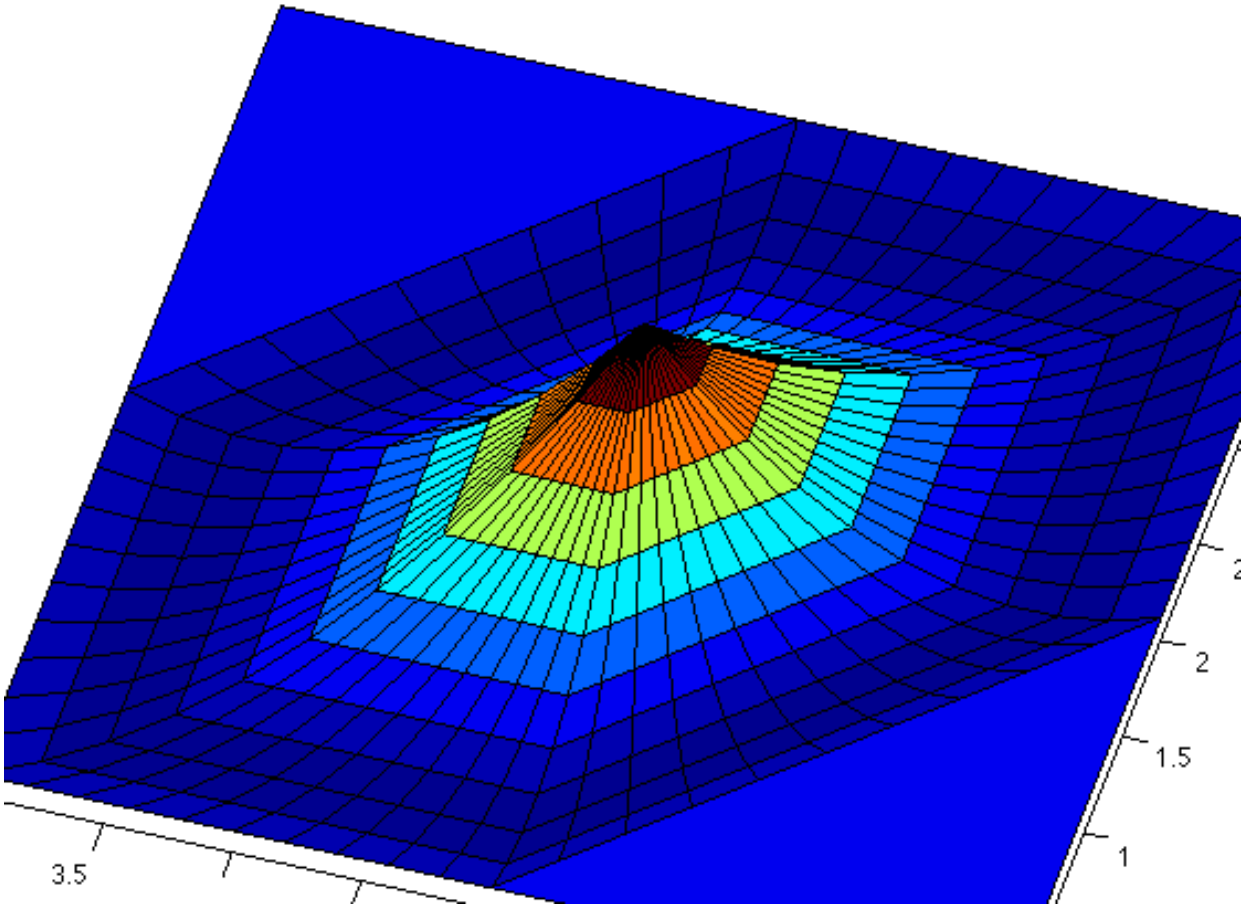


This midside node basis function extends over two elements.

---

http://people.sc.fsu.edu/~jburkardt/m_src/fem2d_basis_t6_display/fem2d_basis_t6_display.html

FEM: A Quadratic Basis Function



This vertex basis function extends over six elements.

http://people.sc.fsu.edu/~jburkardt/m_src/fem2d_basis_t6_display/fem2d_basis_t6_display.html

FEM: Multiply by Test Functions

We have represented u, v and p in terms of basis functions $\psi()$ and $\phi()$, and coefficient vectors a, b, c.

To try to determine the coefficients in these representations, we multiply the state equations by the appropriate test functions:

$$\psi_i \cdot (-\nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + \frac{\partial p}{\partial x}) = f_u$$

$$\psi_i \cdot (-\nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) + \frac{\partial p}{\partial y}) = f_v$$

$$\phi_i \cdot (\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}) = 0$$

FEM: Integrate Over the Region

We integrate each equation over the region Ω:

$$\int_\Omega \psi_i \left(-\nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \frac{\partial p}{\partial x}\right) dx\, dy = f_u$$

$$\int_\Omega \psi_i \left(-\nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) + \frac{\partial p}{\partial y}\right) dx\, dy = f_v$$

$$\int_\Omega \phi_i \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dx\, dy = 0$$

FEM: Integrate By Parts / Green's Theorem

We can solve more general problems if we lower the smoothness requirement on u and v. We can do this using integration by parts:

$$\int_\Omega \nu\left(\frac{\partial \psi_i}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial u}{\partial y}\right) + \psi_i \frac{\partial p}{\partial x}\, dx\, dy = \int_\Omega \psi_i f_u\, dx\, dy + \int_\Gamma \psi_i \frac{\partial u}{\partial n}\, ds$$

$$\int_\Omega \nu\left(\frac{\partial \psi_i}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial v}{\partial y}\right) + \psi_i \frac{\partial p}{\partial y}\, dx\, dy = \int_\Omega \psi_i f_v\, dx\, dy + \int_\Gamma \psi_i \frac{\partial v}{\partial n}\, ds$$

$$\int_\Omega \phi_i \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dx\, dy = 0$$

The right hand sides are only "interesting" (nonzero) for nodes on the boundary where a normal inflow or outflow condition is allowed.

We can still recognize the original PDE's, but the integration allows us to think about the average behavior over the area of each element, rather than at particular points.

FEM: We actually have a matrix problem now!

Essentially, these are the discrete equations we want our computer to solve. The unknown quantities are the coefficient vectors a, b, c used to form u, v and p. Because the equations are linear in the variables, they are linear in the coefficients as well.

And that means that we are actually staring at a very fancy form of the usual linear algebra problem

$$Ax = b$$

Before we can say our problem has become trivial, we have some remaining issues:

- How do we evaluate the basis functions ϕ_i and ψ_i?
- How do we evaluate u, v and p?
- How do we evaluate the integrals that define A and b?

- Equations of Fluid Motion
- A Finite Element Formulation
- **Computing Basis Functions**
- Assembling the Matrix

BASIS: Reference Triangle $\leftrightarrow$ Physical Triangle

Our linear system involves integrals of basis functions and their derivatives. We can approximate the integrals using a **quadrature rule**, a set of n points (x_i, y_i) and weights w_i with which we approximate integrals by

$$I(f, \Omega) = \int_{\Omega} f(x, y)\, dx\, dy \approx \sum_{i=1}^{n} w_i\, f(x_i, y_i) = Q(f, \Omega)$$

The integral approximations can be carried out one element at a time, so we can focus on the problem of estimating an integral over an arbitrary triangle T.

We do this by referring to the unit triangle U formed by vertices $(0, 0), (1, 0), (0, 1)$.

BASIS: The Linear Basis Functions

The basis functions for pressure are defined on the three vertex triangle $T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$. Basis $\phi_1(x, y)$ is 1 at vertex 1, 0 at the other two vertices, and linear over T.

Rather than looking up a formula, can we work one out?

If $\phi_1(x, y)$ is linear, and it's zero at nodes 2 and 3, then it's zero on the line between them. The slope of the line through (x_2, y_2) is:

$$s(x_3, y_3) = \frac{y_3 - y_2}{x_3 - x_2}$$

and for an arbitrary point (x, y), the slope is:

$$s(x, y) = \frac{y - y_2}{x - x_2}$$

We want $\phi_1(x, y)$ to be zero if $s(x, y) = s(x_3, y_3)$. Let's try

$$\phi_1(x, y) \stackrel{?}{=} s(x, y) - s(x_3, y_3) = \frac{y - y_2}{x - x_2} - \frac{y_3 - y_2}{x_3 - x_2}$$

BASIS: The Linear Basis Functions

Let's avoid fractions by multiplying through by the denominators:

$$\phi_1(x, y) \stackrel{?}{=} (y - y_2)(x3 - x_2) - (y_3 - y_2)(x - x_2)$$

Notice that $\phi_1(x2, y2) = \phi_1(x3, y3) = 0$. What more do we need? Oh yes, we need that $\phi_1(x1, y1) = 1$

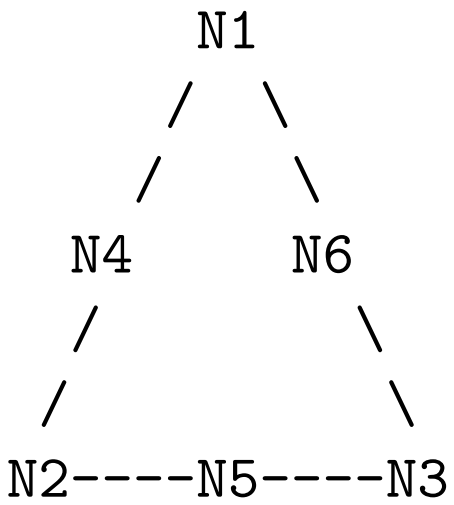
Easy! Just normalize this function by its value at (x_1, y_1):

$$\phi_1(x, y) \stackrel{\checkmark}{=} \frac{(y - y_2)(x3 - x_2) - (y_3 - y_2)(x - x_2)}{(y1 - y_2)(x3 - x_2) - (y_3 - y_2)(x1 - x_2)}$$

Since 1, 2, 3 are "arbitrary", we also defined $\phi_2(x, y)$ and $\phi_3(x, y)$!

BASIS: The Quadratic Basis Functions

Let's symbolize the six node triangle this way:



Just as for the linear basis functions, we can find a linear function which is zero along any line we choose. Therefore, there is a linear function that is zero at N2 and N1 (and hence at N4 as well). Another linear function is zero at N5 and N6, and so on.

Will this help us find a quadratic basis function?

BASIS: The Quadratic Basis Functions

Suppose we want to find ψ_3? There is a linear function $g(x, y)$ that is zero at N1, N4, and N2. There is a linear function $h(x, y)$ that is zero at N5 and N6. Therefore, what about

$$\psi_3(x, y) \stackrel{?}{=} g(x, y)\, h(x, y)$$

Almost, but we need it to be 1 at (x_3, y_3). Easy again:

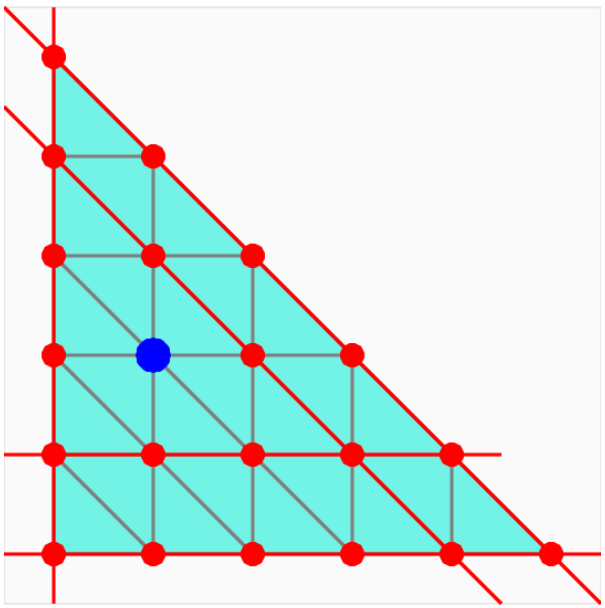
$$\psi_3(x, y) \stackrel{\checkmark}{=} \frac{g(x, y)\, h(x, y)}{g(x_3, y_3)\, h(x_3, y_3)}$$

The product of two linear functions is, of course, quadratic.

Pick any node on the six node triangle, and you can cover the other five nodes with two straight lines. End of story!

BASIS: A Quintic Basis For Triangles

By the way, higher order basis functions are easy. To define a 5-th degree basis function, we simply need five linear factors; that's the same as five straight lines that cover all the other nodes!



To find the linear factors, simply "walk" to each boundary, and note the parallel lines you cross.

$$\psi(x,y) = (x)(y)(y-0.2)(x+y-1)(x+y-0.8)$$

- Equations of Fluid Motion
- A Finite Element Formulation
- Computing Basis Functions
- **Assembling the Matrix**

ASSEMBLY

When it's time to assemble the matrix, we have to keep in mind that we have three variables to worry about and two related grids.

To assemble the equation associated with a variable at a given node, we have to consider all the elements that include that node, all the nodes in those elements, and all the variables associated with those nodes. You can see there can be a lot of bookkeeping!

But at some point, we're looking at the equation for node I, and considering contributions from variables defined at node J. These contributions get added to the (I, J) matrix element, and if we want to, we can call this element A(I,J) for horizontal velocity, B(I,J) for vertical velocity, and C(I,J) for pressure.

ASSEMBLY: The Pressure Equation

The i-th pressure equation (continuity equation) is

$$\int_\Omega \phi_i \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) dx\, dy = 0$$

Every node J neighboring node I contributes to the matrix:

$$A(I,J) = A(I,J) + \int_\Omega \phi_i \frac{\partial \psi_j}{\partial x}\, dx\, dy$$

$$B(I,J) = B(I,J) + \int_\Omega \phi_i \frac{\partial \psi_j}{\partial y}\, dx\, dy$$

ASSEMBLY: The Horizontal Velocity Equation

The left hand side of the i-th horizontal velocity equation is:

$$\int_\Omega \nu\left(\frac{\partial \psi_i}{\partial x}\frac{\partial u}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial u}{\partial y}\right) + \psi_i \frac{\partial p}{\partial x}\, dx\, dy$$

Every node J neighboring node I contributes to A(I,J) and B(I,J):

$$A(I,J) = A(I,J) + \int_\Omega \nu\left(\frac{\partial \psi_i}{\partial x}\frac{\partial \psi_j}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial \psi_j}{\partial y}\right) dx\, dy$$

$$B(I,J) = B(I,J) + \int_\Omega \nu\left(\frac{\partial \psi_i}{\partial x}\frac{\partial \psi_j}{\partial x} + \frac{\partial \psi_i}{\partial y}\frac{\partial \psi_j}{\partial y}\right) dx\, dy$$

and if J is a pressure node, node J also contributes to C(I,J):

$$C(I,J) = C(I,J) + \int_\Omega \psi_i \frac{\partial \phi_j}{\partial x}\, dx\, dy$$

ASSEMBLY: MATLAB Code

```
%
%  Add terms to the horizonal momentum equation.
%
          a(iu,ju) = a(iu,ju) + w(quad) * nu ...
            * ( dbidx(test) * dbjdx(basis) + dbidy(test) * dbjdy(basis) );

          if ( 0 < jp )
            a(iu,jp) = a(iu,jp) + w(quad) * bi(test) * dqjdx(basis);
          end
%
%  Add terms to the vertical momentum equation.
%
          a(iv,jv) = a(iv,jv) + w(quad) * nu ...
            * ( dbidx(test) * dbjdx(basis) + dbidy(test) * dbjdy(basis) );

          if ( 0 < jp )
            a(iv,jp) = a(iv,jp) + w(quad) * bi(test) * dqjdy(basis);
          end
%
%  Add terms to the continuity equation.
%
          if ( 0 < ip )
            a(ip,ju) = a(ip,ju) + w(quad) * qi(test) * dbjdx(basis);
            a(ip,jv) = a(ip,jv) + w(quad) * qi(test) * dbjdy(basis);
          end
```

http://people.sc.fsu.edu/~jburkardt/m_src/fem2d_stokes_sparse/fem2d_stokes_sparse.m

ASSEMBLY: It All Ends Up as a Linear System

Of course, we don't have separate matrices called A, B and C, so we have to store all these coefficients in one big matrix, and we store the coefficients of the representations for u, v and p in one big vector.

Because we have multiple equations and variables, and a pair of grids, a lot of the programming involves simply figuring out where to put things and how to get them back!

We still have some boundary conditions to take care of, but that's another side issue. In the end, we wind up with a sparse linear system:

$$A\,x = b$$

that we solve for the finite element coefficients that give us functional representations of the state variables.

ASSEMBLY: Time Dependent Problems

Our simplified system dropped the time derivatives. If we are interested in the time-dependent behavior of the system, we can approximate time derivatives by finite differences like $\frac{\Delta u}{\Delta t}$.

Our solution scheme must now keep track of two versions of the state variables, u, v, p, namely the "old" or current values, and the "new" values that we must solve for.

In the simplest case, we evaluate most of the equation at the current time, so that at time step k we have a system like:

$$\frac{x^{k+1} - x^k}{\Delta t} + A x^k = b$$

This is easily rewritten as an explicit formula for x^{k+1}. Note that we don't solve a linear system involving A, we multiply by A!

ASSEMBLY: Time Dependent Problems

In the implicit version of the equations, the finite element equations are written "in the future".

$$\frac{x^{k+1} - x^k}{\Delta t} + A x^{k+1} = b$$

This allows larger time steps, but requires the solution of a linear system. The linear system is related to the original "steady state" finite element system, but is, roughly speaking, modified by adding terms involving x^{k+1} in the approximated time derivative.

It turns out that this is even a little more complicated than it sounds, and I hope that Hans-Werner van Wyk will go over methods for handling this kind of computation.

ASSEMBLY: Nonlinear Problems

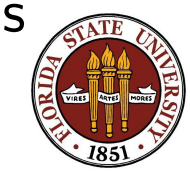
To work with the real Navier-Stokes equations, we need to restore the nonlinear term and solve it. We no longer have a linear system for the discrete variables, but rather something like this:

$$A\,x + g(x) = b$$

Our natural response is to think of the equations as a nonlinear function, and compute the Jacobian:

$$\begin{aligned} F(x) &= b - A\,x - g(x) \\ F'_{i,j} &= \frac{\partial f_i}{\partial x_j} = -A - \frac{\partial g_i}{\partial x_j} \end{aligned}$$

Given a starting point (perhaps by solving the Stokes equations), we can then try Newton's method to get a solution. Note that, as the Reynolds number increases, the problem becomes "more nonlinear" and the Newton iteration may fail to converge.



CONCLUSION: The Big Picture

Navier-Stokes equations govern blood flow and the Gulf Stream.

We can talk about the (almost) automatic solution of the Navier Stokes equations because of new, powerful, free software:

- DEAL.II (C++ based)
- FEniCS (C++ or Python)
- FreeFem++ (C++)
- IFISS (Matlab)

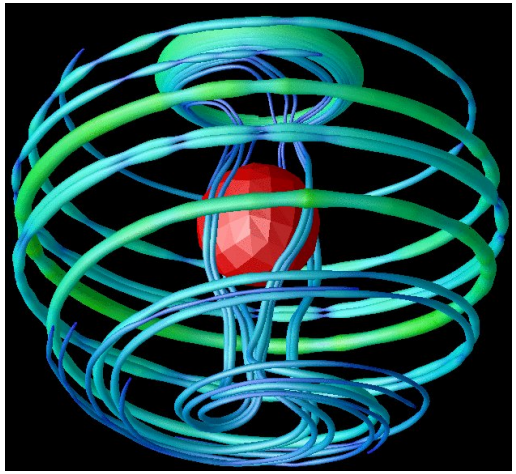
We can:

- see a flow change as we vary the shape of the region, or place obstructions in its path;
- compute the lift of a wing over a range of angles and speeds;
- design streamlined cars or rockets;
- investigate cooling systems for homes or computer centers.

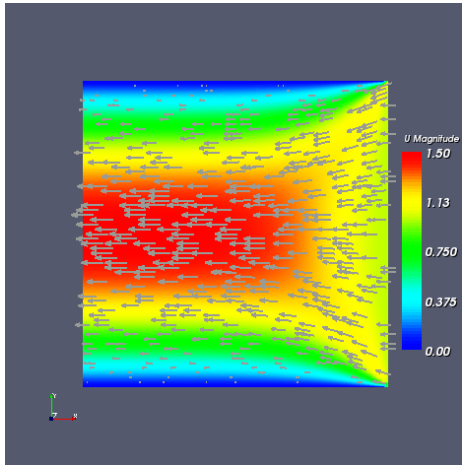
CONCLUSION: Finite Element Flow Solvers

- Howard Elman, Alison Ramage, David Silvester, *IFISS, A Matlab Toolbox for Modeling Incompressible Flow*, ACM Transactions on Mathematical Software, Volume 33, Number 2, June 2007, Article 14.
- Frederic Hecht, *New development in FreeFem++*, Journal of Numerical Mathematics, Volume 20, Number 3-4, 2012, pages 251-265.
- Anders Logg, Kent-Andre Mardal, Garth Wells, *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book,* Lecture Notes in Computational Science and Engineering, Springer, 2011.
- Wolfgang Bangerth, Ralf Hartmann, Guido Kanschat, *DEAL.II - a general-purpose object-oriented finite element library,* ACM Transactions on Mathematical Software, Volume 33, Number 4, article 24, August 2007.

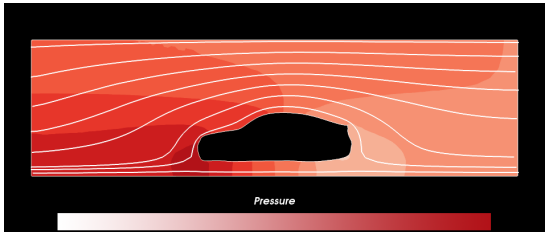
CONCLUSION: The Big Picture



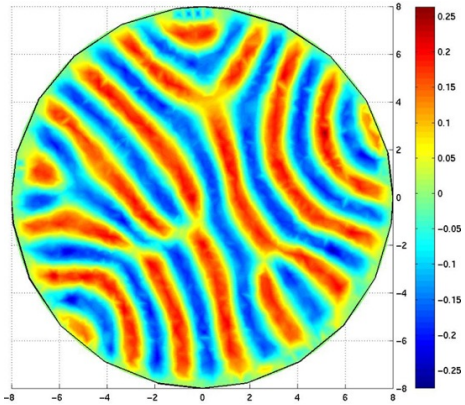
Deal.II



Fenics



FreeFem++



Ifiss