Software packages and FreeFem++

For simple straight forward problems where one has a good idea of what the answer is, there is much to be said for using a pre-existing software package to find the exact answer. Many research problems are however not straight forward and one may have little idea of what the answer is. In these cases, it is certainly better to write one's own tailor-made code that can treat carefully the tricky points.

There are many good software packages for CFD, some free. Propriety packages include COMSOL and ANSYS FLUENT. Free packages include GERRIS a volume of fluid code, OPENFOAM a finite volume code, and FREEFEM++ a finite element code. I do not have a favourite: over the years, I have given lecture demonstrations of several.

Many software packages are easy to use, despite coming with thick manuals. This lecture is a quick introduction of one particular package, FREEFEM++. My aim is to show that packages are not difficult to use to produce useful results.

FREEFEM++ was pioneered in 1987 by Olivier Pironneau in the Laboratoire Jacques-Lious Lion in Paris, and is currently managed by Frédéric Hecht. The finite element package for 2D and 3D includes a simple but versatile mesh generator, around 40 types of finite elements, visualisation of results, MPI parallisation capabilities, various linear solvers including sparse solvers, and more. There is a nice Integrated Development Environment FREEFEM++-CS, which presents three windows, one for the code, one for figures of the results and one for the runtime commentary. There are versions for Linux, Windoze, and Mac OS, all of which are free and can be downloaded from the web. There is a 400 page manual and good tutorial through worked examples.

FreeFem++

1 Poisson problem

Given ρ, to find ϕ:

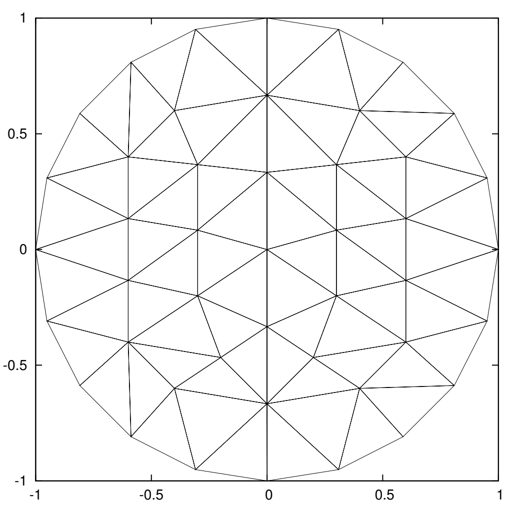
$$\nabla^2\phi = \rho \quad \text{in} \quad r \le 1,$$

$$\phi = 0 \quad \text{on} \quad r = 1.$$

Weak form: multiply by an arbitrary test function w, integrate over domain, reduce high derivatives by integrating by parts using boundary condition,

$$\int_{r\le 1} \nabla\phi\cdot\nabla w + \rho w = 0$$

```
// define boundary - domain to left
border Gamma(t=0,2*pi) { x = cos(t); y = sin(t); }

// construction of mesh of T_h = triangles size h, 20 points on boundary
mesh Th=buildmesh(Gamma(20));

// plot the grid
plot(Th);
```



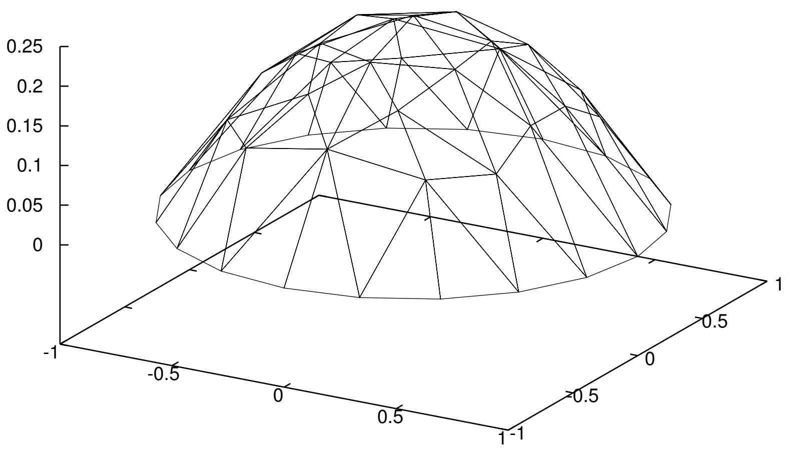
```
// Finite Element space V_h of P1 (linear) elements over mesh Th
fespace Vh(Th,P1);

// unknown phi & test function w over FE space
Vh phi,w;

// set rho(x,y) = -1.0
Vh rho = -1.0;

// solve weak form Poisson equation with BC
solve Poisson (phi,w) = int2d(Th)(dx(phi)*dx(w) + dy(phi)*dy(w))
                      + int2d(Th)(rho*w)
                      + on(Gamma,phi=0);
// bilinear (phi & w) and linear (w only) in separate integrals

// plot the result
plot(phi);
```

See next page for figure.

```
// write to console phi(0,0), should be -0.25
x=0; y=0;
cout << phi << endl;
```

The result for linear P1 elements and 20 points around the boundary is 0.2496. However this is unrepresentively accurate, because with 14 points the value is 0.2545 and with 28 points 0.2484. With linear elements the value at a fixed point can be erratic, depending on how near the point is to a vertex of a triangle. Using quadratic P2 elements, the error at a fixed point decreases quadratically with the mesh size. With 20 points around the boundary, the value at the centre is 0.2455, a 2% error.

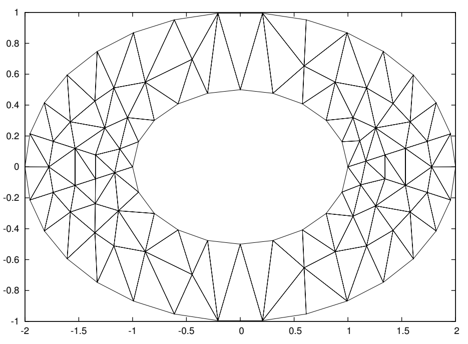
Change to elliptical boundary with elliptical hole.

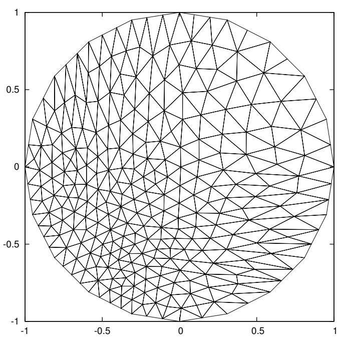


```
// the length of the semimajor axis and  semiminor axis
real a=2.,b=1.;
border Gamma1(t=0,2*pi)    { x = a * cos(t); y = b*sin(t); }
border Gamma2(t=0,2*pi) { x = 0.5*a * cos(t); y = 0.5*b*sin(t); }

// construction of mesh
mesh Th=buildmesh(Gamma1(30)+Gamma2(-20)); //-20 makes hole
plot(Th)
```

Although of little use in this Poisson problem, one can adapt mesh to place more points where a function f large.

```
border Gamma(t=0,2*pi) { x = cos(t); y = sin(t); }
mesh Th=buildmesh(Gamma(20));
plot(Th);
fespace Vh(Th,P1);

func f = 3*(x-0.5)^3 + 3*(y-0.5)^3;
Vh fh = f;

mesh Th2 = adaptmesh(Th,fh);
plot(Th2);
```





2 Driven cavity

Implicit, with pressure penalty

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\delta t}+\mathbf{u}^{n}\cdot\nabla\mathbf{u}^{n}=-\nabla p^{n+1}+\nu\nabla^{2}\mathbf{u}^{n+1},$$

$$\nabla\cdot\mathbf{u}^{n+1}=-10^{-6}p^{n+1}.$$

Weak form: multiply first by $\mathbf{v}$ and second by q, integrate over domain, reduce high derivatives:

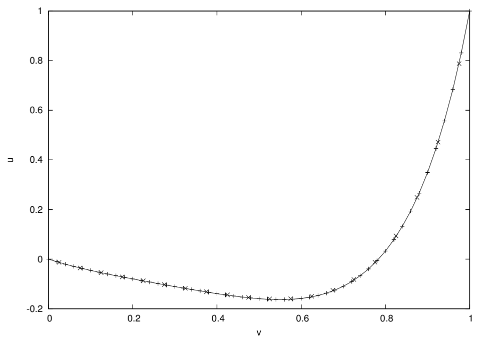
$$\int\frac{1}{\delta t}\mathbf{u}^{n+1}\cdot\mathbf{v}+\nu\nabla\mathbf{u}^{n+1}:\nabla\mathbf{v}-p^{n+1}\nabla\cdot\mathbf{v}-\nabla\cdot\mathbf{u}^{n+1}q-10^{-6}p^{n+1}q$$
$$-\frac{1}{\delta t}\mathbf{u}^{n}\cdot\mathbf{v}+\mathbf{u}^{n}\cdot\nabla\mathbf{u}^{n}\cdot\mathbf{v}\quad=\quad 0.$$

Streamlines by solving Poisson problem

$$\nabla^{2}\psi=-\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}.$$

```
real  nu=1.0/10.0;           // i.e. Re=10
int nx = 20;                 // number of points on a side
real finaltime = 3.0;
int itt = 30;                // number of time steps to stop
real dt=finaltime/itt;
int j;
int i=0;

border bottom (t=0,1.0) {x=t; y=0.0; }
border rside  (t=0,1.0) {x=1.0; y=t; }
border top    (t=0,1.0) {x=1.0-t; y=1.0; }
border lside  (t=0,1.0) {x=0.0; y=1.0-t; }

mesh Th=buildmesh(bottom(nx)+rside(nx)+top(nx)+lside(nx));
plot(Th);

fespace Xh(Th,P2);          // quadratic P2 elements for velocity
fespace Mh(Th,P1);          // linear P1 elements for pressure
Xh ux,uy,  vx,vy,  uxold,uyold;
Mh p,q;

// define Navier-Stokes problem, solve later
//   init=i to store stiffness matrix and not recompute
problem  NS ([ux,uy,p],[vx,vy,q],solver=Crout,init=i) =
    int2d(Th)( (1/dt)*(ux*vx + uy*vy)
               + nu*( dx(ux)*dx(vx) + dy(ux)*dy(vx)
               +  dx(uy)*dx(vy) + dy(uy)*dy(vy) )
               - p*(dx(vx)+ dy(vy))
               - (dx(ux) + dy(uy))*q
               - 0.000001*p*q )
  + int2d(Th)( -(1/dt)*(uxold*vx + uyold*vy)
               + (uxold*dx(uxold) + uyold*dy(uxold))*vx
               + (uxold*dx(uyold) + uyold*dy(uyold))*vy )
  + on(top,ux=sin(pi*x)*sin(pi*x),uy=0)
  + on(lside,bottom,rside,ux=0,uy=0) ;

for (i=0;i<=itt;i++)
 { uxold=ux;      // store old time step,
   uyold=uy;
   NS;  }         // solve NS

// calculate force on top plate as intgral of shear-rate
     cout << "t = " << dt*i << " Force = " <<
        int1d(Th,top)(dy(ux))   << endl;   }
```

```
// output file of horizontal velocity on centreline x=0.5
 ofstream ofile("res.txt");
    for (j=0;j<51;j++)
   { x=0.5;y=j*0.02;
     ofile << y << "   " << ux << endl;  };

// find streamfunction
Xh psi,w;

solve streamlines(psi,w) =
      int2d(Th)( dx(psi)*dx(w) + dy(psi)*dy(w))
   +  int2d(Th)( -w*(dy(ux)-dx(uy)))
   +  on(top,lside,bottom,rside,psi=0);

plot(psi);
```



With 20 points along each of the sides, the value of the steady force on the top surface is found to be

$$F = 3.893.$$