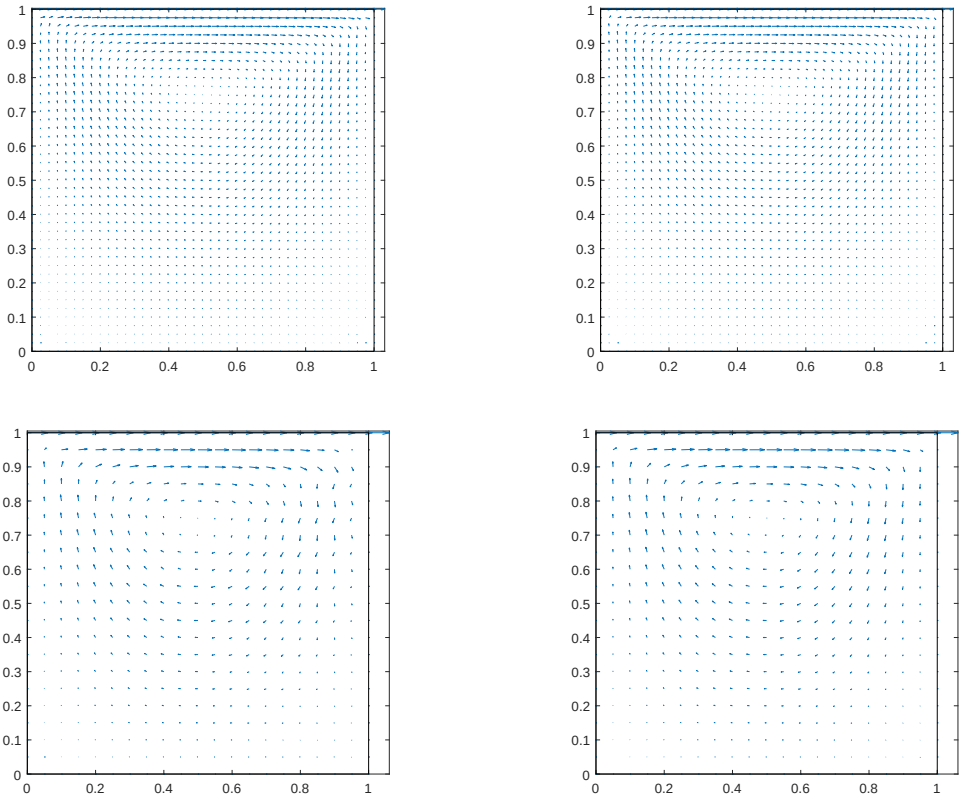
Finite Elements in Fluids-2017. Homework 2: 'Incompressible, viscous flow'

Mauricio Barrera C.

June 5, 2017

The solution of the lid driven cavity flow modeled by a Stokes problem was carried out using the following mixed finite elements: Q_2Q_0 (quadrilateral, second order v, zeroth order p), Q_2Q_1 (quadrilateral, second order v, first order p), P_1P_1 (triangular, first order v, first order p) and $P_1^+P_1$ (triangular, first order plus bubble node v, first order p). The streamline plots are set out below:



Lid-driven cavity flow results: (from left to right and from top to bottom): Q_2Q_0, Q_2Q_1, P_1P_1 and $P_1^+P_1$.

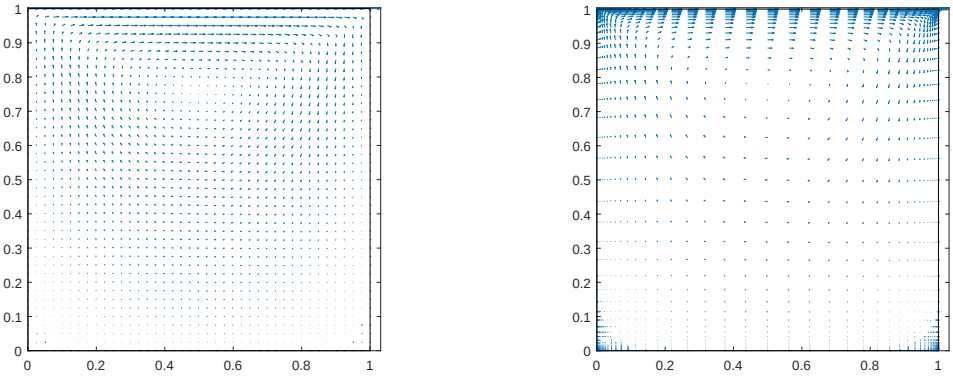
Mathematically, the Stokes problem does not consider the time derivative; thus, the shown results correspond to the physical evolution of the system in time as it tends to infinity. Also, the absence of the convection term implies there is no cause for the flow being turbulent. In this sense, we can be assured that these results actually correspond to a stationary state, so the streamlines are not changing with time either.

The motion in the problem studied here is determined by the moving lid, the no-slip boundary condition and the transport due to diffusive mechanisms. Therefore, the steepest gradients in the output variables are expected to occur close to the boundaries. Indeed, this is reflected by the closeness among streamlines near the cavity walls, particularly just beneath the moving lid.

From the point of view of the elements used, we see that the order of interpolation functions for v is greater than or equal to that of p for all four cases. This is a sufficient but not necessary condition for the Stokes system to be solvable (i.e. non singular coefficient matrix). Satisfaction of the so called LBB condition (Donea and Huerta, 2003) is a necessary condition for said solvability. However, the test is not trivial, and here we can say that, without error messages from the Matlab code

at the time of execution, we can safely say that all produced solvable Stokes systems. However, it is worth highlighting that the P_1P_1 element, where the interpolation orders are the same and there is no aid of a bubble node, may yield a Stokes system with a large condition number: this means that the system could be closer to being singular than when said system comes from application of any of the other elements considered. Even though the computer might have produced the result shown, it might not be as stable or dependable as the others.
Another suitable comparison among elements may lay in element performace, both in terms of convergence and accuracy. Numerical experiments should be carried out for each of the cases; a simple test consists on comparing execution tiems as the mesh experiences some refinement. This would allow, at the vey least, establishing a comparison among the set of elements considered here.
To somehow assess their accuracy, we could compare the results with an element whose good performance is well documented; in this sense we can take the Q_2Q_1 element as reference (one of the so-called Taylor and Hood elements). Hence we can say that element Q_2Q_0 is relatively accurate whereas P_1P_1 and $P_1^+P_1$ underestimate the velocity values, according the the streamlines being more spaced. This may be due to the velocity being interpolated with quadratic polinomials in the Q_2Q_0 case. In addition, the Q_2Q_1 and Q_2Q_0 elements seem to better represent finer details of the flow, such as the sharper streamline contours near the top right corner of the cavity; this feature of the flow could be related to some vorticity phenomena which, should we consider the convective terms, would indicate a potential zone for the emergence of turbulent flow.

When introducing mesh refinement close to the boundaries, and using the Q_2Q_1 element, we have the following results (the overall number of elements stays the same):



Lid-driven cavity flow results, Q_2Q_1 element: without mesh refinement (left) and with mesh refinement close to the boundaries (right).

Even though the stream lines look certainly closer to the boundaries in the case of mesh refinement, this could stem from the fact of having more (nodal velocity) results in this zone, precisely due to the mesh refinement pattern. This coincides with the physical fact of having higher velocity gradients close to a no-slip wall due to the boundary layer taking place there. Nontheless, it is clear that in any flow subject to no slip boundary conditions, the zone close to the boundaries will always experience relatively large velocity gradients, so that accurate simulations demand mesh refinements on these areas.
One way to asses the overall improved accuracy resulting from this mesh refinement could be calculating the mass flow across a vertical line running through the middle of the cavity domain, by computing an expression of the form:

$$\iint_A v_x dA.$$

The net volumetric flux should equal zero on accounts of mass conservation.

It was said earlier that the P_1P_1 element might be close to yield a singular Stokes system, for the interpolation order for velocity and pressure is the same. One strategy to circumvent the non-singluar criterion for Stokes system is adding a stabilisation term. One way of achieving this is the so called Galerkin-Least Squares (GLS).
According to (Hugues and Franca, 1987), cited by (Donea and Huerta, 2003), when we use linear interpolation functions, the modification to the Galerkin statement of the Stokes problem consists on incorporating the following term to the Stokes system:

$$\tau(\nabla q^h, \nabla p^h)$$

where q^h is the discrete weight function acting upon pressure terms, p^h is the discrete approximation function to the pressure

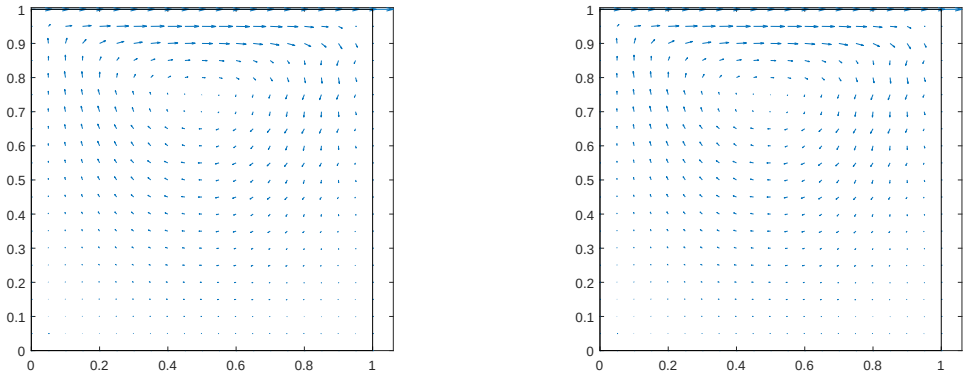
and τ is the stabilisation parameter. According to (Donea and Huerta, 2003), the following expression may be used:

$$\tau = \frac{1}{3}\frac{h^2}{4\nu}$$

where h is a characteristic element size (taken here as $\frac{1}{20}$) and ν is the kinematic viscosity (taken as one in the coming results).
In the Stokes system:

$$\begin{pmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{pmatrix}$$

the zero submatrix is substituted by $\nabla \mathbf{N}_q^T \tau \nabla \mathbf{N}_p$, where $\mathbf{N}_q$ and $\mathbf{N}_p$ are the matrix of discrete weight and approximation functions for pressure, which are taken to be the same in this case.
The streamline plots for $P1P1$ elements with and without GLS stabilisation are set out below:



Lid-driven cavity flow results, P_1P_1 element: with GLS stabilisation (left) and without (right).

As the simulation using P_1P_1 element without GLS stabilisation indeed yielded a unique result, as shown earlier in this report, the similarity of the above results means that the advantage of incorporating the stabilisation term may lie in the condition number of the Stokes system, rather than the fact of the machine producing an answer sucessfully. Insofar as the bottom-left zero submatrix is changed by a non-zero set of entries by the GLS approach, the Stokes system is forced to be nonsingular. It is reasonable to expect, therefore, that the condition number should decrease as a result.

In regards of the provided matlab function StokesSystem.m, in addition to provisions for accomodating the stabilisation term $\nabla \mathbf{N}_q^T \tau \nabla \mathbf{N}_p$, the following lines were added to compute the gradient of approximation functions for pressure, using the available results for the derivatives of velocity shape functions N_x and N_y

```
% Gradient for p
    Nxp=Nx(1,[1 3 5]);
    Nyp=Ny(1,[1 3 5]);
    GradP=[Nxp;Nyp];
```

Then the stabilisation term is calculated as:

```
Ge0 = Ge0 - tau*((GradP)'*GradP)*dvolu;
```

Finally, we solve the Navier-Stokes problem using the provided Matlab file mainNavierStokes.m. The additional convective term:

$$c(\mathbf{w}, \mathbf{v}, \mathbf{v}^*) = \iiint_V \mathbf{w} \cdot (\mathbf{v}^* \cdot \nabla)\mathbf{v} dV$$

is written, for implementation purposes, as shown in (Donea and Huerta, 2003):

$$c_e = \iiint_V N_a \mathbf{v}^h \cdot \nabla N_b dV.$$

This term must fit into the modified Stokes system:

$$\begin{pmatrix} \mathbf{K}+\mathbf{C} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{pmatrix}$$

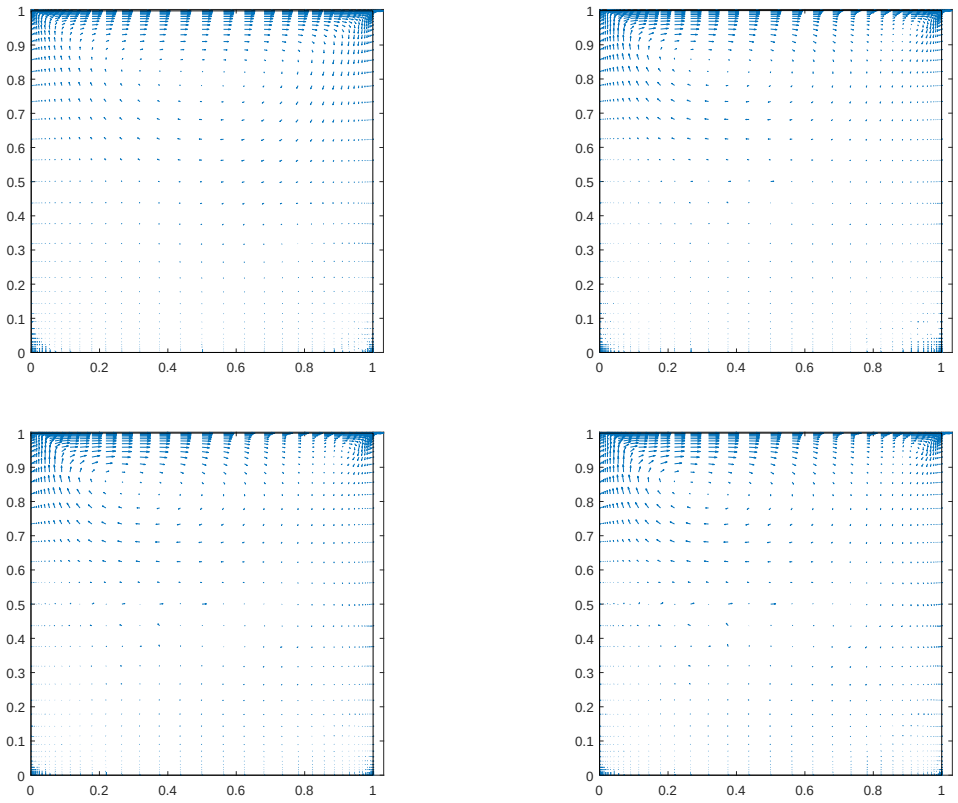
where $\mathbf{C}$ is the matrix expression for the convection term c_e. We see that it must have the same size as $\mathbf{K}$, which, at the element level, is a square matrix of size 18×18 (the Q_2Q_1 element has nine nodes where each stores two components of velocity). In addition, there is a dot product indicating operation by components. The x and y components of ∇N_b are of size 2×18 each (i.e. $[\nabla N_b]_{4\times 18} = [[N_x]_{2\times 18}; [N_y]_{2\times 18})$, so the term $N_a\mathbf{v}^h$ must be composed of two matrices each of size 2×18; each of these must act upon the x and y components of ∇N_b. In this case we have $[N_a]_{2\times 18}$ and $\mathbf{v}^h = diag(v_i), i = 1:18$.

The code for implementing this operation is a Matlab function ConvectionMatrix.m within which a subfunction ElematConv.m performs the actual setup of the aforementioned matrix $\mathbf{C}$. These are set out below:

```
function C = ConvectionMatrix(X,T,XP,TP,velo,referenceElement)

elem = referenceElement.elemV;
ngaus = referenceElement.ngaus;
wgp = referenceElement.GaussWeights;
N = referenceElement.N;
Nxi = referenceElement.Nxi;
Neta = referenceElement.Neta;
NP = referenceElement.NP;
ngeom = referenceElement.ngeom;

% Number of elements and number of nodes in each element
[nElem,nenV] = size(T);
nenP = size(TP,2);

% Number of nodes
nPt_V = size(X,1);
if elem == 11
    nPt_V = nPt_V + nElem;
end
nPt_P = size(XP,1);

% Number of degrees of freedom
nedofV = 2*nenV;
nedofP = nenP;
ndofV = 2*nPt_V;
ndofP = nPt_P;

C=zeros(ndofV,ndofV);

% Loop on elements
for ielem = 1:nElem
    % Global number of the nodes in element ielem
    Te = T(ielem,:);
    TPe = TP(ielem,:);
    % Coordinates of the nodes in element ielem
    Xe = X(Te(1:ngeom),:);
    % Degrees of freedom in element ielem
    Te_dof = reshape([2*Te-1; 2*Te],1,nedofV);
    TPe_dof = TPe;

    Ce = ElematConv(Xe,ngeom,nedofV,nedofP,ngaus,wgp,N,Nxi,Neta,NP,velo,Te_dof);
    C(Te_dof,Te_dof) = C(Te_dof,Te_dof) + Ce;
end

function Ce = ElematConv(Xe,ngeom,nedofV,nedofP,ngaus,wgp,N,Nxi,Neta,NP,velo,Te_dof)
Ce = zeros(nedofV,nedofV);

% Loop on Gauss points
for ig = 1:ngaus
```

```
    N_ig    = N(ig,:);
    Nxi_ig  = Nxi(ig,:);
    Neta_ig = Neta(ig,:);
    NP_ig = NP(ig,:);
    Jacob = [
        Nxi_ig(1:ngeom)*(Xe(:,1)) Nxi_ig(1:ngeom)*(Xe(:,2))
        Neta_ig(1:ngeom)*(Xe(:,1)) Neta_ig(1:ngeom)*(Xe(:,2))
        ];
    dvolu = wgp(ig)*det(Jacob);
    res = Jacob\[Nxi_ig;Neta_ig];
    nx = res(1,:);
    ny = res(2,:);

Ngp = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];
    % Gradient
    Nx = [reshape([1;0]*nx,1,nedofV); reshape([0;1]*nx,1,nedofV)];
    Ny = [reshape([1;0]*ny,1,nedofV); reshape([0;1]*ny,1,nedofV)];
    % Divergence
    dN = reshape(res,1,nedofV);
    N_igx = zeros(2,18);N_igx(1,1:2:17)=N_ig;
    N_igy = zeros(2,18);N_igy(2,2:2:18)=N_ig;
    N_igxy=zeros(2,18)+N_igx+N_igy;
    veloc = velo(Te_dof);
    Dvelo = diag (veloc);
    %Ce = Ce +((N_igx*Dvelo)'*Nx+(N_igy*Dvelo)'*Ny)*dvolu;
    Ce = Ce +((N_igxy*Dvelo)'*Nx+(N_igxy*Dvelo)'*Ny)*dvolu;
end
```

The results for some Reynolds numbers flow are shown next, from left to right and top to bottom, for Re100, Re500, Re1000 and Re1100:



The number of iterations these simulations took are: 10 for Re100; 12 for Re500; 50 for Re1000; and 99 for Re1100. In the latter case the number of iterations reached the maximum set value within the script mainNavierStokes.m. However,

simulations were carried out in intervals of 100 from one Reynolds number to the next, for the results of one simulation constituted the initial guess for the calculations of the next above.

It is clear thas as the Re number goes up, there is a more marked vortex settling on the top left and right corners of the cavity. This makes sense since Re is given by the top lid speed and enforcement of no-slip boundary conditions. Furthermore, in higher Re flows there is dominance of inertial forces, so it is more difficult for the system to reach a steady state. This explains the higher number of iterations needed in such simulations.