CME358 - Assignment 4

Due date: 06/02/2009

Jean-Frédéric Gerbeau (`gerbeau@stanford.edu`)

This document is provided with a directory containing various matlab files available on the course website. These files will be presented in class on May 21st. The following scripts have to be modified and e-mailed to Paolo Massimi `<pmassimi@stanford.edu>`:

- stokesP2P1.m
- stokesP2P0.m
- stokesuzawa.m
- transientstokesuzawa.m

1 P2/P1 and P2/P0 finite elements

The matlab scripts `stokesP2P1.m` and `stokesP2P0.m` respectively implement the P2/P1 and P2/P0 finite elements to solve the stationary Stokes equations

$$\left\{\begin{array}{rcll} -\nu\Delta\boldsymbol{u}+\boldsymbol{\nabla}p &=& \boldsymbol{f} & \text{in } \Omega \\ \operatorname{div}\boldsymbol{u} &=& 0 & \text{in } \Omega \\ \boldsymbol{u} &=& 0 & \text{on } \partial\Omega \end{array}\right.$$

in the primal formulation. The corresponding algebraic system is

$$\begin{bmatrix} A & G \\ D & 0 \end{bmatrix}\begin{bmatrix} U \\ P \end{bmatrix}=\begin{bmatrix} F \\ 0 \end{bmatrix}.$$

To study the convergence rate, it is convenient to build an analytical solution. Consider $\Omega = (0,1)\times(0,1)$ and the scalar functions $V(x,y) = [xy(1-x)(1-y)]^2$, $\Phi(x,y) = \frac{5}{2}y^2 - 10x$ and $g = \nu\text{curl}\,(\textbf{curl}\,V)$. Then the solution of the above problem with $\boldsymbol{f} = \boldsymbol{\nabla}\Phi + \textbf{curl}\,g$ is given by $\boldsymbol{u} = \textbf{curl}\,V$ and $p = \Phi + C^{st}$.

1. The implementation of the right-hand side F is missing in your scripts. Implement it using the mass matrix M.
2. Study the convergence rate of the velocity in the L^2 norm for both pairs of elements. Explain the results.

   Comment: consider a few different values for the discretization step h (e.g. $h = 0.3, \ldots, 0.01$) and plot the error versus h in log-log scale.
3. Explain the difference in the assembling of G for the two pairs of elements.

2 Uzawa algorithm

The matlab script `stokesuzawa.m` computes the block matrices A, G, D corresponding of the stationary Stokes problem.

1. Implement the Uzawa algorithm using the steepest gradient method and the conjugate gradient method provided in the appendix. Consider again the analytical solution given above and plot a norm of the error between the exact solution and the iterates for both methods. Comment. In the remaining, only the conjugate gradient will be used.

   Comments: Do not compute any inverse matrices. Use a tolerance `tol` $= 10^{-6}$.

2. Discuss the dependence of the convergence rate with respect to the discretization step.

3. Modify the block matrix A and the right-hand side F in order to implement the Euler scheme for the transient Stokes problem: let $\delta t > 0$ be the time step,

$$\left\{ \begin{array}{rcl} \boldsymbol{u}^{n+1} - \nu \delta t \Delta \boldsymbol{u}^{n+1} + \delta t \boldsymbol{\nabla} p^{n+1} & = & \boldsymbol{u}^n + \delta t f, \\ \operatorname{div} \boldsymbol{u}^{n+1} & = & 0. \end{array} \right.$$

4. Use the Uzawa algorithm to solve a transient problem (for example with the same force as in the previous questions and starting from a zero initial velocity). Consider different time steps or viscosity ν and comment the convergence rate of the Uzawa algorithm.

Appendix

A Gradient algorithm

```
Real normb = norm(b);
Vector r = b - A*x;
if (normb == 0.0) normb = 1;

resid = norm(r) / normb;

if (resid <= tol) {
    tol = resid;
    max_iter = 0;
    return 0;
 }

for (int i = 1; i <= max_iter; i++) {
    d = r;
    q = A*d;
    alpha = dot(r,r)/ dot(d, q);
    x = x + alpha * d;
    r = r - alpha * q;
    resid = norm(r) / normb;
    if ( resid <= tol) {
      tol = resid;
      max_iter = i;
      return 0;
    }
    rho_1 = rho;
 }
  tol = resid;
```

B Conjugate Gradient algorithm

Adapted from *Iterative Methods Library* (J. Dongarra, A. Lumsdaine, R. Pozo, K. Remington).

```
Real normb = norm(b);
Vector r = b - A*x;
if (normb == 0.0) normb = 1;

resid = norm(r) / normb;

if (resid <= tol) {
    tol = resid;
    max_iter = 0;
    return 0;
 }

for (int i = 1; i <= max_iter; i++) {
    rho = dot(r, r);
    if (i == 1) d = r;
    else d = r + rho / rho_1 * d;
    q = A*d;
    alpha = rho / dot(d, q);
    x = x + alpha * d;
    r = r - alpha * q;
    resid = norm(r) / normb;
    if ( resid <= tol) {
      tol = resid;
      max_iter = i;
      return 0;
    }
    rho_1 = rho;
  }

  tol = resid;
```