Second programming Assignment

Peter Monk

October 23, 2009

You may work in pairs on this assignment (or alone if you prefer). Please download the Navier-Stokes notes from Sakai. Your job is to implement this algorithm in the special case when we neglect the convection terms. So if $\vec{u} = (u, v)^T$ we will be solving the Stokes system

$$\begin{aligned} \frac{\partial \vec{u}}{\partial t} + \nabla p &= \frac{1}{Re}\Delta \vec{u} \\ \nabla \cdot \vec{u} &= 0 \end{aligned}$$

on a simple unit square so $\Omega = [0,1] \times [0,1]$. In addition we will impose the following, somewhat artificial, "driven cavity" boundary conditions

$$\begin{aligned} u = v = 0 &\text{ when } x = 0 \\ u = v = 0 &\text{ when } x = 1 \\ u = v = 0 &\text{ when } y = 0 \\ u = \bar{u},\ v = 0 &\text{ when } y = 1 \end{aligned}$$

where $\bar{u}$ is a given constant and is specified as `ubar` in my code.

Download my code `main.m` which lays out the steps for solving this problem. First is sets the necessary constants (change `imax=jmax=10` for testing your code). I hope that the variables in the code correspond in an obvious way to those in the book. For example `delx` corresponds to δx, and `re` is the Reynolds number. If any variable is not clear to you, e-mail me.

You have to fill in the major steps in the algorithm:

1. First you need to initialize u, v and p by writing a function

   ```
   function [u,v,p]= init_uvp ( imax, jmax, ui, vi);
   ```

   that sets u to be `ui` and v to be `vi` that are already defined in my code. You should set the initial pressure to $p = 0$.

   Here you will encounter an annoying fact that matlab starts its indices at 1 and the book starts at 0. But even more tricky is that we do not need u values at $x = -\Delta x$ (i.e. on the leftmost vertical boundary of Fig 3.5) (this would be denoted $u_{-1,j}$ in the book's notation). Similarly we don't need $u_{imax+1,j}$ (using the book's zero based numbering). The upshot is that you can set

   ```
   u=ui*ones(imax+1,jmax+2)
   ```

   with a similar though reverse initialization for v. Meanwhile the pressure is associated with the center of all cells in the mesh so p is an $imax + 2 \times jmax + 2$ array of zeros. Then the matlab variable $u(1, j + 1)$ corresponds to $u_{0,j}$ in the notes

2. Next you need to set the boundary conditions on u, v by writing a function

   ```
   function [u,v]= setbcond ( u, v,  imax, jmax, ubar);
   ```

   that implements the no-slip conditions (3.21) and (3.23) except on $y = 1$ where using the books numbering you set

   $$u_{i,jmax+1} = 2\bar{u} - u_{i,jmax}, \; i = 1, \vdots, imax$$

3. Next you need to implement the computation of F and G from (3.36) and (3.37)

   ```
    function   [f,g]=comp_fg ( u, v, imax, jmax, delt, delx, dely, re )
   ```

   At the external boundary set $F = u$ (on the right and left boundary) and $G = v$ on the upper and lower boundary.

4. Now we compute the right hand side for the pressure calculation (i.e. the right hand side (3.38))

   ```
   function    rhs= comp_rhs ( f, g, imax, jmax, delt, delx, dely );
   ```

5. Now you can implement SOR for solving (3.38) using a modified version of your Poission code from the first assignment. You will need to use (3.48) within the outer SOR loop (i.e. before each sweep).

   ```
   function    [p,res,iter] = poisson ( p, rhs, imax, jmax, delx, dely,...
    tol, itermax, omega)
   ```

   Stop the SOR method when the relative least squares norm of the residual scaled by the number of mesh points is less than `tol`. More precisely, before starting the SOR loop, compute the least squares norm of the initial guess

   $$P0 = \sqrt{\frac{\sum_{i=1}^{imax} \sum_{j=1}^{jmax} p_{i,j}^2}{imax\, jmax}}$$

   If $P0$ is less than 0.0001, set $P0 = 1$.

   After the SOR iteration, compute the residual $r_{i,j}$ as usual and then compute the relative norm of the residual by

   $$R = \frac{1}{P0} \sqrt{\frac{\sum_{i=1}^{imax} \sum_{j=1}^{jmax} r_{i,j}^2}{imax\, jmax}}$$

   Stop of $R < tol$.

6. Finally you can (3.34) and (3.35) to compute a new u and v within the function

   ```
   function     [u,v]=adap_uv (u, v, f, g, p, imax, jmax, delt, delx, dely )
   ```

You can download my code

```
function plotuv(u,v,p,delx,dely,imax,jmax,t)
```

When you are reasonably sure the code is working, run the code when $imax = jmax = 50$ and plot the final velocity field and pressure field. Hand in the code and the pictures and he output from the code as it steps towards the solution.

See me if you are concerned that you have not computed a “reasonable” solution.