1.723 HW5

Sachith Dunatunga

April 2, 2015

1 Problem 1

1.1 Part 1

The steady-state flow equation is given by

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

where

$$\mathbf{u} = -\lambda(x,y)\nabla p. \tag{2}$$

For a single cell with center at (i,j), we want to integrate

$$\int_{\Omega_{i,j}} \nabla \cdot \mathbf{u} \, \mathrm{dV} = 0 \tag{3}$$

This is converted to a flux from all the surfaces $(i+1/2,j)$, $(i-1/2,j)$, $(i,j+1/2)$, and $(i,j-1)$ via the divergence theorem, yielding

$$\int_{\Gamma_{i+1/2,j}} \mathbf{u}\cdot\mathbf{n} \, \mathrm{dS} + \int_{\Gamma_{i-1/2,j}} \mathbf{u}\cdot\mathbf{n} \, \mathrm{dS} + \int_{\Gamma_{i,j+1/2}} \mathbf{u}\cdot\mathbf{n} \, \mathrm{dS} + \int_{\Gamma_{i,j-1/2}} \mathbf{u}\cdot\mathbf{n} \, \mathrm{dS} = 0, \tag{4}$$

which can be expanded to

$$\int_{\Gamma_{i+1/2,j}} u^x_{i+1/2,j}(1) \, \mathrm{dS} + \int_{\Gamma_{i-1/2,j}} u^x_{i-1/2,j}(-1) \, \mathrm{dS} + \int_{\Gamma_{i,j+1/2}} u^y_{i,j+1/2}(1) \, \mathrm{dS} + \int_{\Gamma_{i,j-1/2}} u^y_{i,j-1/2}(-1) \, \mathrm{dS} = 0. \tag{5}$$

Define the transmissibilities as

$$T^x_{i+1/2,j} = \bar{\lambda}_{i+1/2,j} \frac{\delta y}{\delta x} \tag{6}$$

$$T^x_{i-1/2,j} = \bar{\lambda}_{i-1/2,j} \frac{\delta y}{\delta x} \tag{7}$$

$$T^y_{i,j+1/2} = \bar{\lambda}_{i,j+1/2} \frac{\delta x}{\delta y} \tag{8}$$

$$T^y_{i,j-1/2} = \bar{\lambda}_{i,j-1/2} \frac{\delta x}{\delta y} \tag{9}$$

where

$$\bar{\lambda}_{i+1/2,j} = 2\left(\lambda^{-1}_{i,j} + \lambda^{-1}_{i+1,j}\right)^{-1} \tag{10}$$

$$\bar{\lambda}_{i-1/2,j} = 2\left(\lambda^{-1}_{i,j} + \lambda^{-1}_{i-1,j}\right)^{-1} \tag{11}$$

$$\bar{\lambda}_{i,j+1/2} = 2\left(\lambda^{-1}_{i,j} + \lambda^{-1}_{i,j+1}\right)^{-1} \tag{12}$$

$$\bar{\lambda}_{i,j-1/2} = 2\left(\lambda^{-1}_{i,j} + \lambda^{-1}_{i,j-1}\right)^{-1}. \tag{13}$$

Using the two point flux approximation, we can write the surface integrals as

$$\int_{\Gamma_{i+1/2,j}} u^x_{i+1/2,j}(1)\ \mathrm{dS} = T^x_{i+1/2,j}(p_{i,j} - p_{i+1,j}) \tag{14}$$

$$\int_{\Gamma_{i-1/2,j}} u^x_{i-1/2,j}(-1)\ \mathrm{dS} = T^x_{i-1/2,j}(p_{i,j} - p_{i-1,j}) \tag{15}$$

$$\int_{\Gamma_{i,j+1/2}} u^y_{i,j+1/2}(1)\ \mathrm{dS} = T^y_{i,j+1/2}(p_{i,j} - p_{i,j+1}) \tag{16}$$

$$\int_{\Gamma_{i,j-1/2}} u^y_{i,j-1/2}(-1)\ \mathrm{dS} = T^y_{i,j-1/2}(p_{i,j} - p_{i,j-1}), \tag{17}$$

which allows us to write the sum of fluxes as

$$\begin{aligned}
&-T^x_{i+1/2,j}p_{i+1,j}\\
&-T^x_{i-1/2,j}p_{i-1,j}\\
&\ (T^x_{i+1/2,j} + T^x_{i-1/2,j} + T^y_{i,j+1/2} + T^y_{i,j-1/2})p_{i,j}\\
&-T^y_{i,j+1/2}p_{i,j+1}\\
&-T^y_{i,j-1/2}p_{i,j-1} = 0.
\end{aligned}$$

Using a global numbering $I = f(i,j)$, e.g. $I = N_x i + j$, we can write this as a system of equations for unknown p_I.

Note that the above equation only applies in the interior of the domain. On the boundaries, we must rederive the equations. Setting the transmissibilities to zero suffices to set no-flow boundary conditions. On boundaries where the pressure is known, we must double the interior cell transmissibility; e.g. on the top right corner of the quarter five point, we set

$$T^x_{N_x+1/2,N_y} = 2\lambda_{N_x,N_y}\frac{\delta y}{\delta x} \tag{18}$$

$$T^y_{N_x,N_y+1/2} = 2\lambda_{N_x,N_y}\frac{\delta x}{\delta y} \tag{19}$$

and the boundary term is added in the load vector

$$b_{N_xN_y} = T^x_{N_x+1/2,N_y}\bar{p} + T^y_{N_x,N_y+1/2}\bar{p}. \tag{20}$$

On the inflow boundary (the bottom left cell), we already know the integrated flux entering the system, so we simply replace those integrals in the sum for that cell.

$$\int_{\Gamma_{i+1/2,j}} u^x_{i+1/2,j}(1)\ \mathrm{dS} = T^x_{i+1/2,j}(p_{i,j} - p_{i+1,j}) \tag{21}$$

$$\int_{\Gamma_{i-1/2,j}} u^x_{i-1/2,j}(-1)\ \mathrm{dS} = -Q/2 \tag{22}$$

$$\int_{\Gamma_{i,j+1/2}} u^y_{i,j+1/2}(1)\ \mathrm{dS} = T^y_{i,j+1/2}(p_{i,j} - p_{i,j+1}) \tag{23}$$

$$\int_{\Gamma_{i,j-1/2}} u^y_{i,j-1/2}(-1)\ \mathrm{dS} = -Q/2, \tag{24}$$

which becomes

$$\begin{aligned}
&-T^x_{i+1/2,j}p_{i+1,j}\\
&\ (T^x_{i+1/2,j} + T^y_{i,j+1/2})p_{i,j}\\
&-T^y_{i,j+1/2}p_{i,j+1} = Q.
\end{aligned}$$

We can achieve this by setting the transmissibilites to zero and adding into the load vector

$$b_1 = Q. \tag{25}$$

1.2 Part 2

Here we show three different permeability fields generated using the same parameters as we are to use for the rest of the problem. They look mostly the same, but the scale is different.

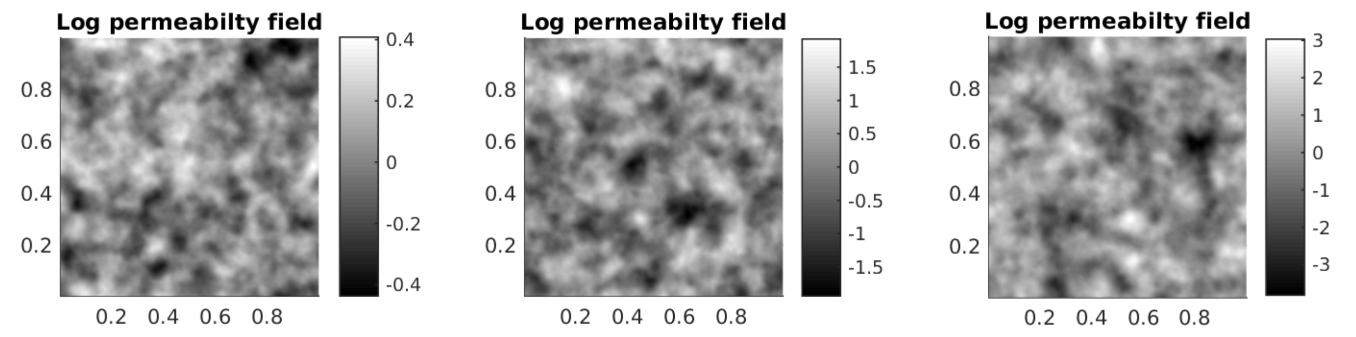

Table 1: Here the variance in log of k varies from 0.1 to 2 to 5 (left to right). The correlation lengths are both $5dx$, where $dx = 1/200$.

Increasing the correlation length in these next images makes larger blocks of similar values.

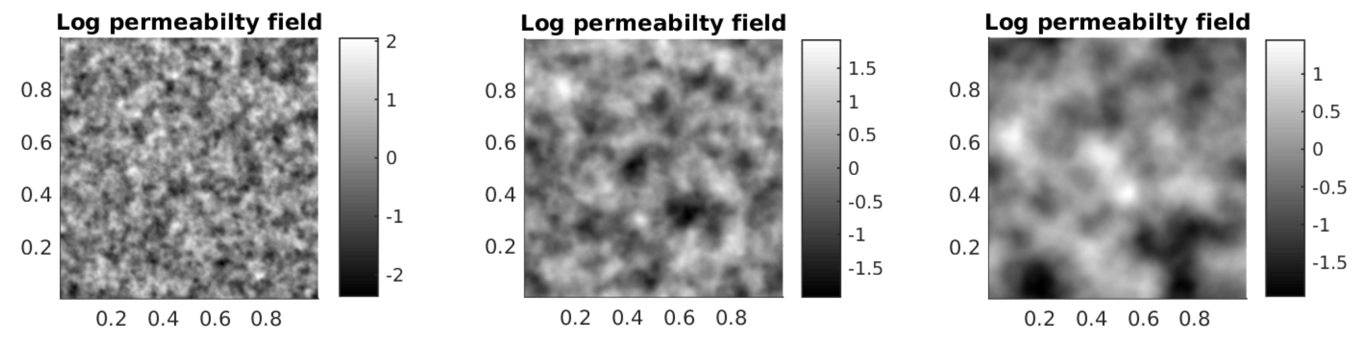

Table 2: Here the variance in log of k is constant at 2. The correlation lengths are equal to each other, but vary from $2dx$, to $5dx$, to $20dx$ from left to right (again $dx = 1/200$).

1.3 Part 3

The code is provided in the zip file, but the main integration routine is reproduced below.

```
function [ Cm, ta, p ] = integrate_scalar_tracer_equation_2d( Nx, Ny, var_lnk, Pe, dt, tf )
%INTEGRATE_SCALAR_TRACER_EQUATION_2D Solves 2D linear scalar transport equation
%using finite volumes.
%  Nx - number of grid cells in x
%  Ny - number of grid cells in x
%  Pe - Peclet number
%  dt - time step size
%  tf - max time
%
%  Returns Nx x Ny x (# times) matrix xm containing cell point locations,
%          Nx x Ny x (# times) matrix cm containing cell concentrations at different times,
%          1 x (# times) matrix ta containing actual times data is saved at.
%           Nx x Ny matrix containing pressure field.

Lx = 1;
Ly = 1;
hx = Lx / Nx;
hy = Ly / Ny;
x = linspace(0+hx/2.0, Lx-hx/2.0, Nx);
y = linspace(0+hy/2.0, Ly-hy/2.0, Ny);
[X,Y] = meshgrid(x,y);
X = X';
Y = Y';

corr_lenx = 4*hx;
corr_leny = 4*hy;

[perm,var_lnk_actual]= random_perm(var_lnk,corr_lenx,corr_leny,Nx,Ny,Lx,Ly);
perm = perm';
kinv = 1./perm;

lambda_bar_x = zeros(Nx+1,Ny);
lambda_bar_x(2:end-1,:) = (hy / hx) * 2 ./ (kinv(2:Nx,:) + kinv(1:Nx-1,:));
lambda_bar_y = zeros(Nx,Ny+1);
lambda_bar_y(:,2:end-1) = (hx / hy) * 2 ./ (kinv(:,2:Ny) + kinv(:,1:Ny-1));


% Set BCs (default is no-flow)
b = zeros(Nx,Ny);
b(1,1) = 1;

lambda_bar_x(end,end) = (hy / hx) * 2 * perm(end,end);
lambda_bar_y(end,end) = (hx / hy) * 2 * perm(end,end);

% make diagonals
T_x_left = reshape(lambda_bar_x(1:end-1, :)', [], 1);
T_x_right = reshape(lambda_bar_x(2:end, :)', [], 1);
T_y_bottom = reshape(lambda_bar_y(:, 1:end-1)', [], 1);
T_y_top = reshape(lambda_bar_y(:, 2:end)', [], 1);

sumT = T_x_right + T_x_left + T_y_bottom + T_y_top;
diags = [-T_x_left, -T_y_bottom, sumT, -T_y_top, -T_x_right];

% assemble matrix
A = spdiags(diags, [Ny;1;0;-1;-Ny], Nx*Ny, Nx*Ny)';
bv = reshape(b, [], 1);

% solve for pressure
pv = A \ bv;
p = reshape(pv, Nx, Ny);

% show plots and save pressure
% figure; surf(X,Y,log(perm),'edgecolor','none'); view([0, 0, 1]);
% figure; surf(X,Y,p,'edgecolor','none'); view([0, 0, 1]);
% h = figure; plot(x, diag(p) - p(end/2,end/2));
% set(h, 'units', 'inches', 'position', [1 1 3 3])
% set(h, 'PaperUnits','centimeters');
% set(h, 'Units','centimeters');
% pos=get(h,'Position');
% set(h, 'PaperSize', [pos(3) pos(4)]);
% set(h, 'PaperPositionMode', 'manual');
% set(h, 'PaperPosition',[0 0 pos(3) pos(4)]);
% fname = sprintf('figs/pressure_%d_%d.png', 10*var_lnk, Nx);
% title(sprintf('Pressure along diagonal\n(N = %d)', Nx));
% xlabel('Coordinate');
% ylabel('Pressure');
% print(fname, '-dpng');

% calculate integrated velocities
ux_int = lambda_bar_x(2:end-1, :).*(p(1:end-1, :) - p(2:end, :));
uy_int = lambda_bar_y(:, 2:end-1).*(p(:, 1:end-1) - p(:, 2:end));
zx = zeros(1, size(ux_int, 2));
zy = zeros(size(uy_int, 1), 1);
ux = [zx; ux_int; zx];
uy = [zy, uy_int, zy];
ux(1,1) = 1/2;
uy(1,1) = 1/2;
ux(end,end) = 1/2;
uy(end,end) = 1/2;

% draw appoximate velocities on perm field
% scale = 0.1;
% figure; h = surf(X,Y,zeros(size(X)), log(perm),'edgecolor','none'); view([0, 0, 1]);
% axis equal square;
% hold on;
% quiver(X,Y,scale*(ux(1:end-1,:)+ux(2:end, :))/2, scale*(uy(:, 1:end-1) + uy(:, 2:end))/2, '
    Autoscale','off', 'color', [0,0,0]);
% hold off;

t = 0;
i = 1;
c = zeros(Nx, Ny);
while (t < tf)
    t = t + dt;
    ta(i) = t;
    Cm(:,:, i) = c;
    c(1,1) = 1;

    Fx_adv = ux_int.*(ux_int > 0).*c(1:end-1, :) + ux_int.*(ux_int < 0).*c(2:end, :);
    zFx = zeros(1, size(Fx_adv, 2));
    Fx_adv = [zFx; Fx_adv; zFx];

    Fy_adv = uy_int.*(uy_int > 0).*c(:, 1:end-1) + uy_int.*(uy_int < 0).*c(:, 2:end);
    zFy = zeros(size(Fy_adv, 1), 1);
    Fy_adv = [zFy, Fy_adv, zFy];

    Fx_diff = (1 / Pe) * (hy / hx) * (c(1:end-1, :) - c(2:end, :));
    Fx_diff = [zFx; Fx_diff; zFx];

    Fy_diff = (1 / Pe) * (hx / hy) * (c(:, 1:end-1) - c(:, 2:end));
    Fy_diff = [zFy, Fy_diff, zFy];

    Fx = Fx_adv + Fx_diff;
    Fy = Fy_adv + Fy_diff;

    Fx(1,1) = 1/2;
    Fy(1,1) = 1/2;
    Fx(end,end) = c(end,end)*1/2;
    Fy(end,end) = c(end,end)*1/2;

    dc = (dt / (hx * hy)) * (Fx(1:end-1,:) - Fx(2:end, :) + Fy(:, 1:end-1) - Fy(:, 2:end));
    c = c + dc;

    % plot concentration
    % surf(X,Y,c, 'edgecolor', 'none');
    % view([0 0 1]);
    % colormap(hot);
    % drawnow;

    i = i + 1;
end

end
```

Listing 1: integrate_scalar_tracer_equation_2d.m

1.4 Part 4

We plot the pressure along the diagonal over multiple grid spacings. For low variance (0.1), we see that the solution looks like that for a point source and a point sink, as expected. When the variance is larger though we see a lot of inhomogeneity both within the solution and also between solutions (the curves don't look like they will converge to anything, although I'm sure there is a 'distribution average'). The pressures themselves seem to be higher with increasing range in permeabilities.

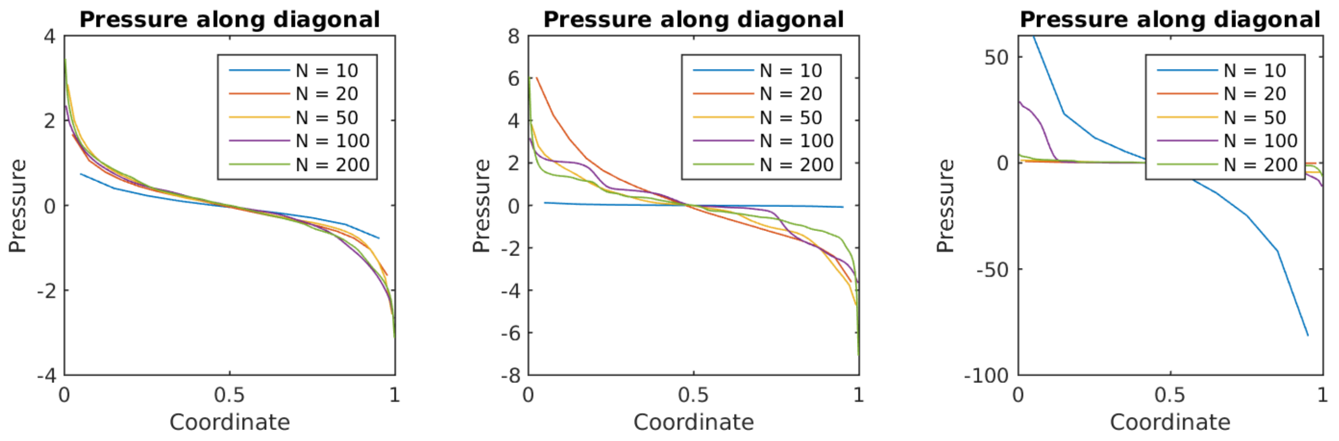

Table 3: The pressure along the diagonal is plotted for multiple grid spacings for various variances in log permeability (0.1, 2, and 5 from left to right).