Homework 6, Problem 4

Table of Contents

This script uses the shooting method to solve a second order BVP Alison Cozad, November 2012

Main script

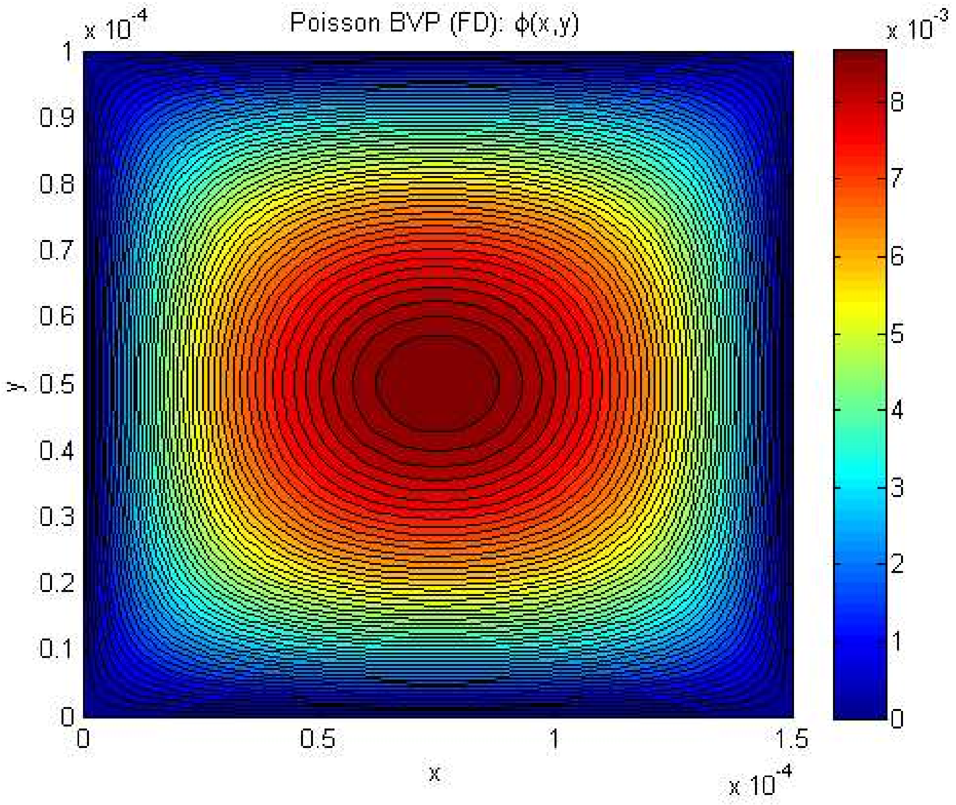
```
function HW6P4

    clc
    clear all
    close all
    clf

    L=150e-6;    % [m]
    H=100e-6;    % [m]
    N=50;        % [#]
```

Part a

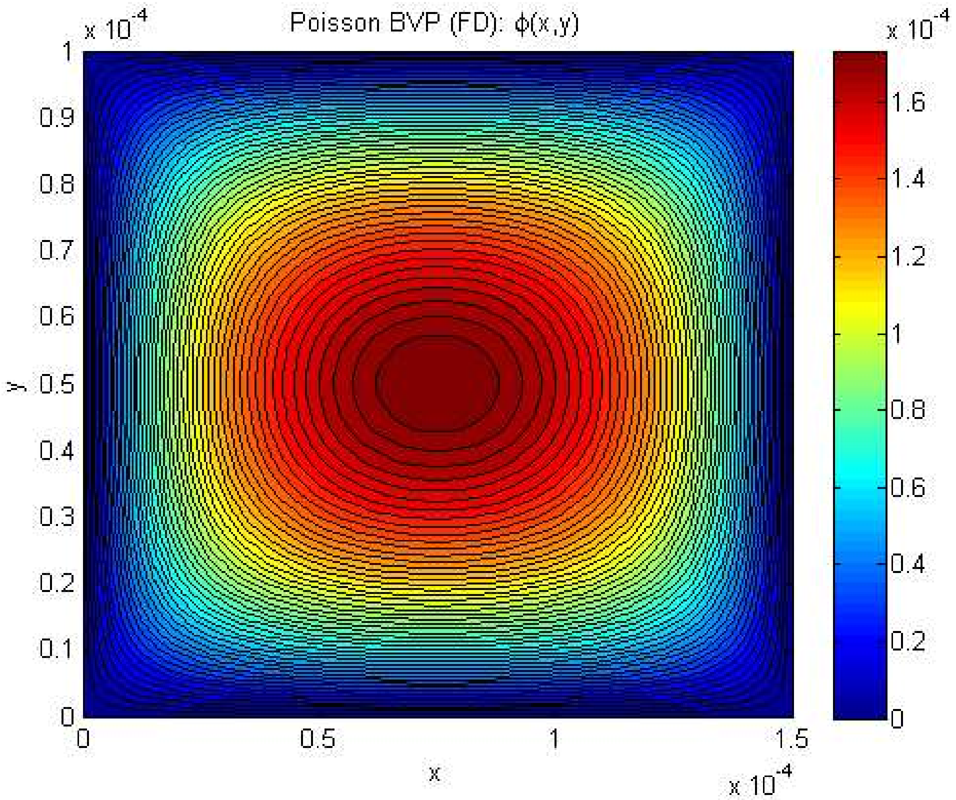
```
    mu=1e-3;     % [Pa*s]
    % Solve BVP
    [xa,ya,phia,iflag_main] = BVP_2D_Poisson_FD(@(x,y,L,H) ...
        sourceTerm(x,y,L,H,mu),L,H,N);
    % Determine volume flowrate
    V=trapz(ya(1,:),trapz(xa(:,1),phia,1),2);
    fprintf('\nPart a: mu=%g Pa*s, V=%g m^3/s or %g mm^3/s\n',mu, ...
        V,V*1e3^3);
    fprintf('                        v=%g m/s or %g mm/s\n', V/(L*H),...
        V/(L*H)*1e3);
```



Part b

```
mu=50*mu;     % [Pa*s]
% Solve BVP
[xb,yb,phib,iflag_main] = BVP_2D_Poisson_FD(@(x,y,L,H) sourceTerm(x,y,...
    L,H,mu),L,H,N);
% Determine volume flowrate
V=trapz(yb(1,:),trapz(xb(:,1),phib,1),2);
fprintf('\nPart b: mu=%g Pa*s, V=%g m^3/s or %g mm^3/s\n',mu, V,...
    V*1e3^3);
fprintf('                        v=%g m/s or %g mm/s\n', V/(L*H),...
    V/(L*H)*1e3);
```

```
    Part b: mu=0.05 Pa*s, V=1.28734e-12 m^3/s or 0.00128734 mm^3/s
                          v=8.58226e-05 m/s or 0.0858226 mm/s
```




```
end
```

Defining the source term

```
function f=sourceTerm(x,y,L,H,mu)
    g = 9.8;          %    [m/s^2]
    rho = 998;        %    [kg/m^3]
    deltaP = 1e3;     %    [Pa/m]

    % Define source term
    f=1/mu*(-deltaP+rho*g);

    % Put into matrix form
    f=f*ones(size(x));
end
```

```
% BVP_2D_Poisson_FD.m
% This MATLAB routine solves a 2-D boundary value
% problem on a rectangular domain using finite differences.
% Zero Dirichlet boundary conditions are used.
% K. Beers. MIT ChE. 9/4/03

% For the CMU homeowrk, I have commented with %%%%AC where I have made
```

```
% changes.  But I have really only changed what this function has output

%%%%AC I have made it so this problem returns the velocity profile
function [XG,YG,PHIG,iflag_main] = ...
    BVP_2D_Poisson_FD(fun_name,L,H,num_pts);
iflag_main = 0;

% set number of points in grid
Nx = num_pts; Ny = num_pts;  Ntot = Nx*Ny;

% First, place a 2-D computational grid.
x = linspace(0,L,Nx); dx = x(2)-x(1);
y = linspace(0,H,Ny); dy = y(2)-y(1);

% Form a 2-D regular mesh such that
%     XG(i,j) = x(i), YG(i,j) = y(j)
[XG,YG] = form_2D_mesh(x,y);

% compute value of f(x,y) at each grid point
% and make a filled contour plot
FG = feval(fun_name,XG,YG,L,H);
%%%%AC This plot actually produce a blank contour plot, so i will comment
%%%%it out
%figure; contourf(XG,YG,FG,min(Nx,Ny)); colorbar;
%xlabel('x'); ylabel('y'); title('f(x,y)');

% allocate memory for the matrix and RHS vector
A = spalloc(Ntot,Ntot,5*Ntot); b = zeros(Ntot,1);

% We next specify equations for each boundary point.
% BC # 1
i = 1;
for j=1:Ny
    n = get_label(i,j,Nx,Ny);
    A(n,n) = 1;  b(n) = 0;
end
% BC # 2
i = Nx;
for j=1:Ny
    n = get_label(i,j,Nx,Ny);
    A(n,n) = 1;  b(n) = 0;
end
% BC # 3
j = 1;
for i=2:(Nx-1)
    n = get_label(i,j,Nx,Ny);
    A(n,n) = 1; b(n) = 0;
end
% BC # 4
j = Nx;
for i=2:(Nx-1)
    n = get_label(i,j,Nx,Ny);
    A(n,n) = 1; b(n) = 0;
end
```

```
% We now set the linear equations for the interior points.
factor_x = 1/(dx^2); factor_y = 1/(dy^2);
factor_cent = 2*(factor_x + factor_y);
for i=2:(Nx-1)
for j=2:(Ny-1)
    n = get_label(i,j,Nx,Ny);
    A(n,n-Ny) = -factor_x;  A(n,n+Ny) = -factor_x;
    A(n,n-1) = -factor_y;  A(n,n+1) = -factor_y;
    A(n,n) = factor_cent;
    b(n) = FG(i,j);
end
end

% We now solve using Gaussian elimination
phi = A\b;

% We now extract the results of the calculation
% into the 2-D grid format and make a filled
% contour plot.
PHIG = zeros(size(XG));
for i=1:Nx
for j=1:Ny
    n = get_label(i,j,Nx,Ny);
    PHIG(i,j) = phi(n);
end
end

figure; contourf(XG,YG,PHIG,min(Nx,Ny)); colorbar;
xlabel('x'); ylabel('y'); title('Poisson BVP (FD): \phi(x,y)');

% save the results to a .mat file
save BVP_2D_Poisson_FD.mat;

iflag_main = 1;
end


%---------------------------------------
function [XG,YG] = form_2D_mesh(x,y);

Nx = length(x); Ny = length(y);
XG = zeros(Nx,Ny); YG = zeros(Nx,Ny);
for k=1:Nx
    XG(k,:) = x(k)*ones(1,Ny);
end
for k=1:Ny
    YG(:,k) = y(k)*ones(Nx,1);
end

end


%---------------------------------------
```

```
function n = get_label(i,j,Nx,Ny);

n = (i-1).*Ny + j;

end
```

```
Part a: mu=0.001 Pa*s, V=6.4367e-11 m^3/s or 0.064367 mm^3/s
                       v=0.00429113 m/s or 4.29113 mm/s
```

Published with MATLAB® 7.14