RBF Assignment Four

Desmond Blue Manthy

1 Report:

By my understanding, the method by which we solve the coupled system:

$$\frac{\partial \omega}{\partial t} = \left\{ \frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} \right\} + \frac{1}{\mathrm{Re}}\Delta\omega \qquad (1)$$

$$\Delta\psi + \omega = 0 \qquad (2)$$

is to create an ω vector and advance it through time by feeding a function to ode45. To generate this function, we need to solve for ψ at every time step based on the given ω. To do this, we discretize $\Delta\ \psi$ using RBF-FD and solve the linear system. Coupling this with an LU factorization of the differentiation matrix of the linear system allows for an efficient method of finding ψ based on the given ω at each time step. This allows us to calculate the various derivatives, again using RBF-FD, and ultimately compute the right hand side (RHS) of equation (1). This is then used to model ω, the vorticity of the node points, over time.

To accomplish this, I first loaded in the interior and boundary and set up nodes and set up ω, a vector containing ω over both the interior and boundary nodes. Then I set the initial conditions for ω by selecting two small regions and assigning them opposite signed vorticity. Next, since the differentiation matrix (DM) of the linear system for solving equation (2) is always the same, I decided to calculate it and then pass in its LU factorization to my ode function as global variables.

I struggled to find the DM for the laplacian of . I understand that I needed to discretize the laplacian and set up a linear system in order to solve equation (2) in which ω is given over the interior and boundary nodes. The DM should have come out to be a massive non symmetrical sparse matrix with no node ordering. Then I should have set the diagonal in the bottom rows to 1 to keep ψ from updating. The size of the DM and it not being simitric requires the use of l,u,p,q factorization. However, I was unable to get this part working, which made testing the remainder of my code difficult. I plan to go back and rework this part after finals. Provided a proper DM and factorization the next step was to create a function to pass to ode45. I called this function odefun, which takes

in ω. In this function the first step was to calculate ψ via the DM factorization asψneeds to be calculated over the interior nodes each timestep based on the updated ω. This cost of this repeated calculation is why we needed to factorize the DM.

After finding ψ for the given time step, I attempted to solve the derivatives on the RHS of equation (1). I began by doing RBF-FD differentiation. I do believe there is something inforrect with my method. For starters the way I am calculating the RBF-FD weights they will be the same at each time step. It makes more sense to me that as the vorticity changes the weights would change as well. This was difficult to investigate without being able to look at the results visually (do to my improper DM). Lastly, I plotted the resulting ω in the same way I plotted ω_0 at the start. With a properly set up DM this would have allowed for looking at the vorticity at various times t.

I want to apologize for not coming in and asking questions about this assignment. I found it very intimidating to ask questions in front of PHD students. I should have asked you questions directly, but I felt embarrassed about not being able to figure this out when the other students seemed to have it under control. The closer the due date got, the more I felt like I should have had more done and that I wasn't far enough along to ask for help. I know I robbed myself of a real opportunity to learn this material more deeply. Especially when I know I struggle at moving from theory to application. If you are willing, I would appreciate being able to take some of your time this summer to ask about this assignment, not for credit towards the class. I think RBFs are an amazing tool and their applications you showed us are amazing and they are a tool I would love to have a solid grasp on.

2 MATLAB Code:

```
% Manthy Assignment 4 RBF
clear; close all

% ---------------------------------------------------------------------------
% Load boundary and interior nodes
int = importdata('Interior_nodes.txt');
ext = importdata('Boundary_nodes.txt');

global int_x
global int_y
int_x = int(:,1);
int_y = int(:,2);
ext_x = ext(:,1);
ext_y = ext(:,2);
for i = ext_x
```

```
    int_x = [int_x;i];
end
for i = ext_y
    int_y =[int_y;i];
end

% disp(int_x)
% disp(int)
% disp(int_y)

global Re
Re = 70; %Reynolds number
global N
N = length(int_x);
dt = 0.1;
endt = 1000;
t = linspace(0,dt,endt);
dt = 0.1;

%set inital conditions
w0 = zeros(776, 1);
% Choose as initial condition two small regions of opposite signed vorticity
x = [-2,-1,0,1,2];
w0(111:115,1) = normpdf(x).';
w0(443:447,1) = -normpdf(x).';
% w0(300:300,1)=-.5;
% w0(150:150,1)=.5;
% disp(w0)

xlin = linspace(min(int_x), max(int_x), 100);
ylin = linspace(min(int_y), max(int_y), 100);
[X,Y] = meshgrid(xlin, ylin);
Z = griddata(int_x, int_y, w0, X, Y);
mesh(X,Y,Z); title('Vorticity'); xlabel('x'); ylabel('y');
axis tight;

%TODO create mesh
%{
tiledlayout('flow')

nexttile
plot(int, '.');
xlabel('x'); ylabel('y'); title('Interior');

nexttile
plot(ext, '.');
```

```
xlabel('x'); ylabel('y'); title('Boundry');
%}
A = [int_x int_y];
% disp(A)

% ---------------------------------------------------------------------------
%Solve equation (2)
%LU Factorization
%Able to find once because the DM matrix is always the same
%Use RBF-FD to discritize the laplacian and make linear system
w = RBF_FD_PHS_pol_weights (int_x,int_y,3,1);
% r = w0 .* w(:,2);
% disp(r)

% disp(w)
% spdiags([8 -1 1 -8 8 -1 1 -8]/12.*ones(N,1), [1-N,2-N,-2,-1,1,2,N-2,N-1], N, N);
DM = sparse(w); %this is incorrect in a lot of ways
% disp(DM)
global L
global U
global P
global Q

%compute factorization
[L,U,P,Q] = lu(DM);
%P*DM*Q = L*U => DM = inv(P)*L*U*inv(Q)

%{
subplot(1,2,1)
nexttile
spy(L)
title('L Factor')
subplot(1,2,2)
nexttile
spy(U)
title('U Factor')
%}

% ---------------------------------------------------------------------------
% ode45
[~,W] = ode45(odefun,t,w0);

% ---------------------------------------------------------------------------
% display results
% need to display surface of non gridbased data
xlin = linspace(min(int_x), max(int_x), 133);
```

```
ylin = linspace(min(int_y), max(int_y), 133);
[X,Y] = meshgrid(xlin, ylin);
Z = griddata(int_x, int_y, W, X, Y);
mesh(X,Y,Z); title('Vorticity'); xlabel('x'); ylabel('y');
axis tight;

% ------------------------------------------------------------------------
% Functions

% s = fcn(2);
% disp(s)

% function a = fcn(b)
% global N
% a = N*b;
% end

%function to be passed to ode45
function f = odefun (w)
%{
This fcn needs to:
    Produce the RHS of (1) given and w vector

Input parameters

w   omega vorticity fcn. a column vector for the unknown omega over the
interior and boundry nodes

%}
global Re
global L
global U
global P
global Q

% Re = 70; %Reynolds number


%solve for psi based on omega
%DM = inv(P)*L*U*inv(Q)
%to solve DM*a=b => inv(P)*L*U*inv(Q)*a=b => a=inv(P)*L*U*inv(Q)\b
B = inv(P)*L*U*inv(Q);
% disp(B)
psi =  B\w;

%solve variouse derivatives
```

```
global int_x
global int_y
weights = RBF_FD_PHS_pol_weights (int_x,int_y,3,1); %#ok<NASGU>

%d psi/ d y
dpdy = psi .* weights(:,2);

%d psi/ d x
dpdx = psi .* weights(:,1);

%d omega / d y
dwdy = w .* weights(:,2);

%d omega / d x
dwdx = w .* weights(:,1);

%laplacian of omega (d2/dx2+d2/dy2)w
lapw = w .* weights(:,3).';

%need to include 0's to keep from updating boundry
f = ((dpdy.*dwdx) -(dpdx.*dwdy)) - (1/Re)*lapw;
end

%Provided fcn for calculating RBF-FD weights in 2-D when using PHS+poly discretization
function w = RBF_FD_PHS_pol_weights (x,y,m,d)
% Calculate RBF-FD weights in 2-D for d/dx, d/dy, and the Laplacian
% d2/dx2+d2/dy2, based on PHS+poly local approximations

% Input parameters
% x,y   Column vectors with locations for the nodes in a single
%       stencil; approximation to be accurate at loction x(1),y(1)
% m     Power of r in RBF fi(r) = r^m, with m odd, >= 3.
% d     Degree of supplementary polynomials (d = -1 no polynomials) %

% Output parameter
% w     Matrix with three columns, containing RBF-FD weights for
%       d/dx, d/dy, and the Laplacian d2/dx2+d2/dy2, respectively.

x = x-x(1);   y = y-y(1);   % Shift nodes so stencil centered at origin
n  = length(x);

% ------ RBF part ------------------------------------------------------
A0 = sqrt((x-x').^2+(y-y').^2).^m; % RBF A-matrix
L0 = m*(hypot(x,y).^(m-2).*[-x,-y,m*ones(n,1)]); % RHSs

% ------ Polynomial part -----------------------------------------------
```

```
if d == -1                          % Special case; no polynomial terms,
     A = A0;  L = L0;                % i.e. pure RBF
else    % Create matrix with polynomial terms and matching constraints
    X   =  x(:,ones(1,d+1));  X(:,1) = 1;  X = cumprod( X,2);
    Y   =  y(:,ones(1,d+1));  Y(:,1) = 1;  Y = cumprod( Y,2);
    np  = (d+1)*(d+2)/2;            % Number of polynomial terms
    XY  = zeros(n,np); col = 1;     % Assemble polynomial matrix block
    for k = 0:d
        XY(:,col:col+k) = X(:,k+1:-1:1).*Y(:,1:k+1);
        col = col+k+1;
    end
    L1 = zeros(np,3);               % Create matching RHSs
    if d >= 1; L1(2,1) = 1; L1(3,2) = 1; end
    if d >= 2; L1(4,3) = 2; L1(6,3) = 2; end
    A = [A0,XY;XY',zeros(col-1)];   % Assemble linear system to be solved
    L = [L0;L1];                    % Assemble RHSs
end

% ------ Solve for weights -------------------------------------------------
W  = A\L;                           % Solve linear system
w  = W(1:n,:);                      % Extract the RBF-FD weights
end
}
```