Appendix B Least-Squares Finite Element Method Source Code

B.1 Main Subroutine

```
 clc
 clear
 close all
% ----------------
% Input parameters
% ----------------
Re = 1;
M = 150;

magnetic_lx = 2.5;
magnetic_ly = -0.2;

% ----------------
% Create mesh
% ----------------
% Import mesh from GMSH
square % The mesh and nodes element details can be referred in B.4

% Assign to new variables
node_xy     = msh.POS(:,1:2);
edge_node   = msh.LINES4;
elem_node   = msh.QUADS12;

% Connectivity matrix
elem_node = elem_node(:,1:end-1);

% Mesh parameter
num_node        = size(node_xy,1);
num_elem        = size(elem_node,1);
num_bnd_edge    = size(edge_node,1);
elem_order      = size(elem_node,2);

% ----------------
% Variables numbering
% ----------------
num_dof = num_node*4;

node_u_var = 1:4:num_dof;
node_v_var = 2:4:num_dof;
node_p_var = 3:4:num_dof;
node_w_var = 4:4:num_dof;

% ----------------
% Boundary condition
% ----------------
% Impose edge boundary [Edge BC flag, BC Value]
[node_u_bc,node_v_bc,node_p_bc,...
 node_u_condition,node_v_condition,node_p_condition]=...
            bc_channel(num_node,node_xy);
```

```
% ----------------------
% Gauss points and weights
% ----------------------

gp = [-0.8611 -0.8611   0.1210;
      -0.8611 -0.33998  49/216;
      -0.8611  0.33998  49/216;
      -0.8611  0.8611   0.1210;
      -0.33998 -0.8611  49/216;
      -0.33998 -0.33998 0.4253;
      -0.33998 0.33998  0.4253;
      -0.33998 0.8611   49/216;
       0.33998 -0.8611  49/216;
       0.33998 -0.33998 0.4253;
       0.33998 0.33998  0.4253;
       0.33998 0.8611   49/216;
       0.8611  -0.8611  0.1210;
       0.8611  -0.33998 49/216;
       0.8611  0.33998  49/216;
       0.8611  0.8611   0.1210];

%   ----------------------
%   Global matrix assembly
%   ----------------------
%   Initialize global matrix and vector
K = zeros(num_dof,num_dof);
F = zeros(num_dof,1);

%   Loop all element - assembly from local to global.
for ie = 1:num_elem

    %   Global node index
    iu = node_u_var(elem_node(ie, :));
    iv = node_v_var(elem_node(ie, :));
    ip = node_p_var(elem_node(ie, :));
    iw = node_w_var(elem_node(ie, :));
    ind=[iu iv ip iw];

    %   Coordinates for local nodes
    x = node_xy(elem_node(ie, :),1);
    y = node_xy(elem_node(ie, :),2);

    %   Initialize local matrix and load vector
        S11 = zeros(elem_order);
        S22 = zeros(elem_order);
        S12 = zeros(elem_order);
        S21 = zeros(elem_order);
        S20 = zeros(elem_order);
        S10 = zeros(elem_order);
        S02 = zeros(elem_order);
        S01 = zeros(elem_order);
        S00 = zeros(elem_order);

        Zero44=zeros(elem_order);
        Zero41=zeros(elem_order,1);

        Gx = zeros(elem_order,1);
        Gy = zeros(elem_order,1);
        Mx = zeros(elem_order,1);
        My = zeros(elem_order,1);
```

```
%   Numerical integration
for ig = 1:size(gp,1) %length(gp)

    % Gauss point weight
    X = gp(ig,1);
    Y = gp(ig,2);
    W = gp(ig,3);

    % Shape function, derivaties and jacobian
    [N, dNdX, dNdY, J, detJ, invJ] = shape_function(x,y,X,Y);

    % Body force
    xn=N*x;
    yn=N*y;

    A=magnetic_lx;
    B=magnetic_ly;

    H = abs(B)/((xn-A)^2 + (yn-B)^2)^(1/2);
    dHdx=-(abs(B)*(xn-A))/(((yn-B)^2 + (xn-A)^2)^(3/2));
    dHdy=-(abs(B)*(yn-B))/(((yn-B)^2 + (xn-A)^2)^(3/2));

    fp = H;
    fx = H*dHdx;
    fy = H*dHdy;

    % Sum the function evaluated at GP
    S11 = S11 + W*(dNdX'*dNdX)*det(J);
    S22 = S22 + W*(dNdY'*dNdY)*det(J);
    S12 = S12 + W*(dNdX'*dNdY)*det(J);
    S21 = S21 + W*(dNdY'*dNdX)*det(J);
    S20 = S20 + W*(dNdY'*N)*det(J);
    S10 = S10 + W*(dNdX'*N)*det(J);
    S02 = S02 + W*(N'*dNdY)*det(J);
    S01 = S01 + W*(N'*dNdX)*det(J);
    S00 = S00 + W*(N'*N)*det(J);

    Gx = Gx + W*(dNdX'*fx)*det(J);
    Gy = Gy + W*(dNdY'*fy)*det(J);

    Mx = Mx + W*(dNdY'*fx)*det(J);
    My = My + W*(dNdX'*fy)*det(J);

    RR11=(S11+S22);
    RR12=(S12-S21);
    RR13=Zero44;
    RR14=S20;

    RR21=S21-S12;
    RR22=S11+S22;
    RR23=Zero44;
    RR24=-S10;

    RR31=Zero44;
    RR32=Zero44;
    RR33=S11+S22;
    RR34=(1/Re)*(S12-S21);

    RR41=S02;
```

```
        RR42=-S01;
        RR43=(1/Re)*(S21-S12);
        RR44=(1/Re^2)*(S11+S22)+S00;
        F11=Zero41;
        F21=Zero41;
        F31=M*(Gx+Gy);
        F41=(M/Re)*(Mx-My);

       k =[RR11 RR12 RR13 RR14 ;
           RR21 RR22 RR23 RR24 ;
           RR31 RR32 RR33 RR34 ;
           RR41 RR42 RR43 RR44 ];

       f = [F11; F21; F31; F41];

    end

    % Assemble local to global
    K(ind, ind) = K(ind, ind) + k;
    F(ind,1)    = F(ind,1) + f;

end

% ----------------------
% Impose boundary conditions
% ----------------------
for i = 1:num_node

    iu = node_u_var(i);
    iv = node_v_var(i);
    ip = node_p_var(i);
    iw = node_w_var(i);

%   Check if node is boundary
    if node_u_condition(i) == 1
        F           = F - K(:,iu) * node_u_bc(i);
        K(iu,:)     = 0;
        K(:,iu)     = 0;
        K(iu,iu)    = 1;
        F(iu)       = node_u_bc(i);
    end
    if node_v_condition(i) == 1
        F           = F - K(:,iv) * node_v_bc(i);
        K(iv,:)     = 0;
        K(:,iv)     = 0;
        K(iv,iv)    = 1;
        F(iv)       = node_v_bc(i);
    end
    if node_p_condition(i) == 1
        F           = F - K(:,ip) * node_p_bc(i);
        K(ip,:)     = 0;
        K(:,ip)     = 0;
        K(ip,ip)    = 1;
        F(ip)       = node_p_bc(i);
    end

end
```

```
% ----------------------
% Solve simultaneous equations
% ----------------------
UV = K\F;

% --------------------
% Plot the result
% --------------------
 % Plot the Mesh
figure;
patch('Faces', elem_node(:,[1 5 6 2 7 8 3 9 10 4 11 12]), ...
    'Vertices', node_xy, 'EdgeColor','black',...
    'FaceColor','none');

% Plot the velocity contour
types = 'Velocity';
figure(2);
[FU,FV,U2,V2,X2,Y2,Total_Velocity]=contourplot2(node_xy,node_u_var,n
ode_v_var,UV,types);

% Plot the streamlines
types = 'Streamline';
figure(3);
[FU,FV,U2,V2,X2,Y2]=streamlineplot(node_xy,node_u_var,node_v_var,UV,
types);
axis([0 5 0 1])
```

B.2 Subroutine to Impose Boundary

```
function[node_u_bc,node_v_bc,node_p_bc,...
         node_u_condition,node_v_condition,node_p_condition]=...
         bc_channel(num_node,node_xy)

tol = 1e-9;
node_u_condition = zeros(1, num_node);
node_v_condition = zeros(1, num_node);
node_p_condition = zeros(1, num_node);

node_u_bc = zeros(1, num_node);
node_v_bc = zeros(1, num_node);
node_p_bc = zeros(1, num_node);

for i=1:num_node

   x = node_xy(i,1);
   y = node_xy(i,2);

%   Inlet - x=0
   if ( abs(x - 0)<= tol )
        node_u_condition(i) = 1;
        node_u_bc(i) = 4*y*(1-y);
```

```
        node_v_condition(i) = 1;
        node_v_bc(i) = 0;
    end

%   Bottom boundary y=0
    if ( abs(y - 0)<= tol )
        node_u_condition(i) = 1;
        node_u_bc(i) = 0;

        node_v_condition(i) = 1;
        node_v_bc(i) = 0;
    end

%   Top boundary y=1
    if ( abs(y - 1)<= tol )
        node_u_condition(i) = 1;
        node_u_bc(i) = 0;

        node_v_condition(i) = 1;
        node_v_bc(i) = 0;
    end

%   Bottom left point x=10, y=0
    if ( abs(x - 5)<= tol andand abs(y - 0)<= tol )
        node_p_condition(i) = 1;
        node_p_bc(i) = 0;
    end

end
end
```

B.3 Subroutine Shape Function

```
function [N, dNdX, dNdY, J, detJ, invJ] = shape_function(x,y,X,Y)

%   ---------------------
%   Shape function Q12
%   ---------------------
    N    = [
              ((X - 1)*(Y - 1)*(9*X^2 + 9*Y^2 - 10))/32
             -((X + 1)*(Y - 1)*(9*X^2 + 9*Y^2 - 10))/32
              ((X + 1)*(Y + 1)*(9*X^2 + 9*Y^2 - 10))/32
             -((X - 1)*(Y + 1)*(9*X^2 + 9*Y^2 - 10))/32
               (9*(Y - 1)*(- 3*X^3 + X^2 + 3*X - 1))/32
              -(9*(Y - 1)*(- 3*X^3 - X^2 + 3*X + 1))/32
              -(9*(X + 1)*(- 3*Y^3 + Y^2 + 3*Y - 1))/32
               (9*(X + 1)*(- 3*Y^3 - Y^2 + 3*Y + 1))/32
               (9*(Y + 1)*(- 3*X^3 - X^2 + 3*X + 1))/32
              -(9*(Y + 1)*(- 3*X^3 + X^2 + 3*X - 1))/32
              -(9*(X - 1)*(- 3*Y^3 - Y^2 + 3*Y + 1))/32
               (9*(X - 1)*(- 3*Y^3 + Y^2 + 3*Y - 1))/32
            ]';
    dNdX = [
```

```
              -((Y - 1)*(- 27*X^2 + 18*X - 9*Y^2 + 10))/32
                -((Y - 1)*(27*X^2 + 18*X + 9*Y^2 - 10))/32
                  ((Y + 1)*(27*X^2 + 18*X + 9*Y^2 - 10))/32
                 ((Y + 1)*(- 27*X^2 + 18*X - 9*Y^2 + 10))/32
                          (9*(Y - 1)*(- 9*X^2 + 2*X + 3))/32
                            (9*(Y - 1)*(9*X^2 + 2*X - 3))/32
                 (27*Y^3)/32 - (9*Y^2)/32 - (27*Y)/32 + 9/32
                 (27*Y)/32 - (9*Y^2)/32 - (27*Y^3)/32 + 9/32
                          -(9*(Y + 1)*(9*X^2 + 2*X - 3))/32
                         -(9*(Y + 1)*(- 9*X^2 + 2*X + 3))/32
                 (27*Y^3)/32 + (9*Y^2)/32 - (27*Y)/32 - 9/32
               - (27*Y^3)/32 + (9*Y^2)/32 + (27*Y)/32 - 9/32
             ]';
    dNdY = [
               -((X - 1)*(- 9*X^2 - 27*Y^2 + 18*Y + 10))/32
                ((X + 1)*(- 9*X^2 - 27*Y^2 + 18*Y + 10))/32
                  ((X + 1)*(9*X^2 + 27*Y^2 + 18*Y - 10))/32
                 -((X - 1)*(9*X^2 + 27*Y^2 + 18*Y - 10))/32
               - (27*X^3)/32 + (9*X^2)/32 + (27*X)/32 - 9/32
                 (27*X^3)/32 + (9*X^2)/32 - (27*X)/32 - 9/32
                         -(9*(X + 1)*(- 9*Y^2 + 2*Y + 3))/32
                          -(9*(X + 1)*(9*Y^2 + 2*Y - 3))/32
                 (27*X)/32 - (9*X^2)/32 - (27*X^3)/32 + 9/32
                 (27*X^3)/32 - (9*X^2)/32 - (27*X)/32 + 9/32
                           (9*(X - 1)*(9*Y^2 + 2*Y - 3))/32
                          (9*(X - 1)*(- 9*Y^2 + 2*Y + 3))/32
             ]';

%   Jacobian
    J    = [dNdX*x dNdX*y; dNdY*x dNdY*y];
    detJ = J(1,1)*J(2,2) - J(1,2)*J(2,1);
    invJ = [J(2,2) -J(1,2); -J(2,1) J(1,1)]/detJ;

%   Derivatives of shape function
    dN = J\[dNdX;dNdY];     % Equivalent to inverse multiplication
    dNdX = dN(1,:);
    dNdX = dN(2,:);
```

B.3 Subroutine to Plot Result on Velocity Contour

```
function
[FU,FV,U2,V2,X2,Y2,Total_Velocity]=contourplot2(node_xy,node_u_var,n
ode_v_var,UV,types)

[X2,Y2] =
meshgrid(min(node_xy(:,1)):0.05:max(node_xy(:,1)),min(node_xy(:,2)):
0.05:max(node_xy(:,2)));

FU = scatteredInterpolant(node_xy(:,1),node_xy(:,2),UV(node_u_var));
FV = scatteredInterpolant(node_xy(:,1),node_xy(:,2),UV(node_v_var));
U2 = FU(X2,Y2);
V2 = FV(X2,Y2);

Total_Velocity=sqrt(U2.^2+V2.^2);
```

```
% Plot the contour
if strcmp(types,'Velocity')
hold on
cla
contourf(X2,Y2,Total_Velocity,'LineStyle','none','Fill','on','ShowTe
xt','on',...
'LevelList',min(min(Total_Velocity)):(max(max(Total_Velocity))-
min(min(Total_Velocity)))/100:max(max(Total_Velocity)))

quiver(X2(1:5:end,1:5:end),Y2(1:5:end,1:5:end),U2(1:5:end,1:5:end),V
2(1:5:end,1:5:end),1,'AutoScale','on','Color','black');
colormap jet
else
end
```

B.4 Subroutine the Mesh that Import from Gmsh

```
% MATLAB mesh
% square, Created by Gmsh
% ASCII
clear msh;
msh.nbNod = 33;
msh.POS = [
0 0 0;
1 0 0;
1 1 0;
0 1 0;
0.4999999999976003 0 0;
0.1666666666659685 0 0;
0.3333333333317844 0 0;
0.6666666666650668 0 0;
0.8333333333325333 0 0;
1 0.4999999999976003 0;
1 0.1666666666659685 0;
1 0.3333333333317844 0;
1 0.6666666666650668 0;
1 0.8333333333325333 0;
0.5000000000026048 1 0;
0.833333333334227 1 0;
0.6666666666683256 1 0;
0.3333333333350699 1 0;
0.1666666666675349 1 0;
0 0.5000000000026048 0;
0 0.833333333334227 0;
0 0.6666666666683256 0;
0 0.3333333333350699 0;
0 0.1666666666675349 0;
0.5000000000001025 0.5000000000001026 0;
0.4999999999984343 0.1666666666667009 0;
0.4999999999992684 0.3333333333334017 0;
0.3333333333334016 0.5000000000009367 0;
0.1666666666667008 0.5000000000017707 0;
0.5000000000009366 0.666666666666735 0;
```

```
0.5000000000017707 0.8333333333333675 0;
0.8333333333333675 0.4999999999984344 0;
0.666666666666735 0.4999999999992685 0;
];
msh.MAX = max(msh.POS);
msh.MIN = min(msh.POS);
msh.LINES4 =[
 1 5 6 7 1
 5 2 8 9 1
 2 10 11 12 2
 10 3 13 14 2
 3 15 16 17 3
 15 4 18 19 3
 4 20 21 22 4
 20 1 23 24 4
];
msh.QUADS12 =[
 1 5 25 20 6 7 26 27 28 29 23 24 1
 20 25 15 4 29 28 30 31 18 19 21 22 1
 5 2 10 25 8 9 11 12 32 33 27 26 1
 25 10 3 15 33 32 13 14 16 17 31 30 1
];
```

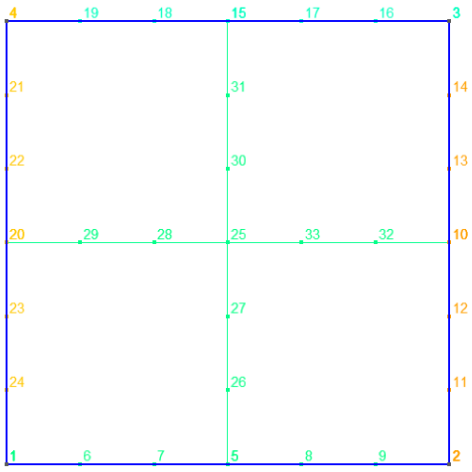

Figure B.1 12-node rectangular element