Contents


```
% HW 6
```

plot mesh

```
clear;close all

% File names
nodf='hw44.nod';
belf='hw44.bel';
dndf='hw44.dnd';

node=load(nodf);
node=node(:,2:3);        % All nodes
bel=load(belf);
bel=bel(:,2:3);          % Boundary element incidence list
bnod=unique(bel(:));     % Boundary nodes
dnd=load(dndf);
dnd=dnd(:,1);            % Type 1 BC nodes (ground)
```

FEM original solution

```
elem = load( 'hw44.ele' );
bcs = load( 'hw44.dnd' );
A_e = zeros( 3 );
A = zeros( size(node,1)-3 );
B = zeros( size(node,1)-3, 1 );

for i=1:size(elem,1)
    n1 = elem(i,2);
    n2 = elem(i,3);
    n3 = elem(i,4);

    x1 = node(n1,1);
    y1 = node(n1,2);
    x2 = node(n2,1);
    y2 = node(n2,2);
    x3 = node(n3,1);
```

```
    y3 = node(n3,2);

    area = (x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2)) / 2;

    t1 = 1 + x1/10 + y1/5;
    t2 = 1 + x2/10 + y2/5;
    t3 = 1 + x3/10 + y3/5;


    A_e(1,1) = -t1 * (((y2-y3)*(y2-y3))+((x2-x3)*(x2-x3))) / (4*area);
    A_e(1,2) = -t1 * (((y2-y3)*(y3-y1))+((x2-x3)*(x3-x1))) / (4*area);
    A_e(1,3) = -t1 * (((y2-y3)*(y1-y2))+((x2-x3)*(x1-x2))) / (4*area);
    A_e(2,1) = -t2 * (((y3-y1)*(y2-y3))+((x3-x1)*(x2-x3))) / (4*area);
    A_e(2,2) = -t2 * (((y3-y1)*(y3-y1))+((x3-x1)*(x3-x1))) / (4*area);
    A_e(2,3) = -t2 * (((y3-y1)*(y1-y2))+((x3-x1)*(x1-x2))) / (4*area);
    A_e(3,1) = -t3 * (((y1-y2)*(y2-y3))+((x1-x2)*(x2-x3))) / (4*area);
    A_e(3,2) = -t3 * (((y1-y2)*(y3-y1))+((x1-x2)*(x3-x1))) / (4*area);
    A_e(3,3) = -t3 * (((y1-y2)*(y1-y2))+((x1-x2)*(x1-x2))) / (4*area);

    A(n1,n1) = A(n1,n1) + A_e(1,1);
    A(n1,n2) = A(n1,n2) + A_e(1,2);
    A(n1,n3) = A(n1,n3) + A_e(1,3);
    A(n2,n1) = A(n2,n1) + A_e(2,1);
    A(n2,n2) = A(n2,n2) + A_e(2,2);
    A(n2,n3) = A(n2,n3) + A_e(2,3);
    A(n3,n1) = A(n3,n1) + A_e(3,1);
    A(n3,n2) = A(n3,n2) + A_e(3,2);
    A(n3,n3) = A(n3,n3) + A_e(3,3);
end
```

Boundary conditions - Dirichlet

```
for i = 1:size(bcs,1)
   B(bcs(i,1)) = bcs(i,2);
   A(bcs(i,1),:) = 0;
   A(bcs(i,1),bcs(i,1)) = 1;
end
```

Sink

```
B(492) = .25;
B(493) = .25;
```

Source

```
n1 = elem(288,2);
n2 = elem(288,3);
n3 = elem(288,4);

x1 = node(n1,1);
y1 = node(n1,2);
x2 = node(n2,1);
y2 = node(n2,2);
```

```
x3 = node(n3,1);
y3 = node(n3,2);

area = (x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2)) / 2;

% source at node 503, goes through 504,505?
B( elem(288,2) ) = -( (x2*y3-x3*y2) + node(503,1)*(y2-y3) - node(503,2)*(x2-x3) ) / (2*area);
B( elem(288,3) ) = -( (x3*y1-x1*y3) + node(503,1)*(y3-y1) - node(503,2)*(x3-x1) ) / (2*area);
B( elem(288,4) ) = -( (x1*y2-x2*y1) + node(503,1)*(y1-y2) - node(503,2)*(x1-x2) ) / (2*area);



Sol = linsolve(A,B);
% figure(1);
% for i = 1:size(elem,1)
%     patch( node(elem(i,2:4),1), node( elem(i,2:4),2), Sol(elem(i,2:4)), 'FaceColor', 'interp
' );
%     hold on;
% end
% colorbar;
% title( 'Original Potential Solution');
```

Inversion

```
pt_x = [-.6, -.3, 0, .3, .6, -.3, 0, .3, .6, 0, .3, .6];
pt_y = [.2, .2, .2, .2, .2, .4, .4, .4, .4, .6, .6, .6];

% find elements where sample points are located
pt_elem = zeros(1,12);
pt_size = size(pt_x,2);
for i=1:pt_size
    x = pt_x(i);
    y = pt_y(i);
    for j=1:size(elem,1)
        [in] = inpolygon( x, y, [node(elem(j,2),1), node(elem(j,3),1), node(elem(j,4),1)], [no
de(elem(j,2),2), node(elem(j,3),2), node(elem(j,4),2)] );
        if in == 1
            pt_elem(i) = j;
        end
    end
end

% construct Sample matrix
S = zeros( pt_size, size(node,1)-3 );
for i=1:pt_size
    n1 = elem(pt_elem(i),2);
    n2 = elem(pt_elem(i),3);
    n3 = elem(pt_elem(i),4);

    x1 = node(n1,1);
    y1 = node(n1,2);
    x2 = node(n2,1);
    y2 = node(n2,2);
    x3 = node(n3,1);
    y3 = node(n3,2);
```

```
    area = (x1*(y2-y3) + x2*(y3-y1) + x3*(y1-y2)) / 2;

    S( i, n1 ) = -( (x2*y3-x3*y2) + pt_x(i)*(y2-y3) - pt_y(i)*(x2-x3) ) / (2*area);
    S( i, n2 ) = -( (x3*y1-x1*y3) + pt_x(i)*(y3-y1) - pt_y(i)*(x3-x1) ) / (2*area);
    S( i, n3 ) = -( (x1*y2-x2*y1) + pt_x(i)*(y1-y2) - pt_y(i)*(x1-x2) ) / (2*area);
end

% Covariance of B
Cov_d = zeros( size(node,1)-3 );
sigma = .1;
l = .4;
for i=1:size(node,1)-3
    for j=1:size(node,1)-3
        r_ij = sqrt( (node(i,1)-node(j,1))^2 + (node(i,2)-node(j,2))^2 );
        Cov_d(i,j) = sigma^2 * (1+r_ij/l) * exp( -r_ij/l );
    end
end

Cov_i = zeros( size(node,1)-3 );
sigma = 1;
l = .3;
for i=1:size(node,1)-3
    for j=1:size(node,1)-3
        r_ij = sqrt( (node(i,1)-node(j,1))^2 + (node(i,2)-node(j,2))^2 );
        Cov_i(i,j) = sigma^2 * (1+r_ij/l) * exp( -r_ij/l );
    end
end

Cov_b = Cov_d + Cov_i;
```

Solve procedure

```
W_d = (.05)^2*eye(pt_size);
for i = 1:pt_size
    delta = zeros(1,pt_size);
    delta(i) = 1;

    lambda = 2*transpose(inv(A))*(transpose(S)*W_d)*transpose(delta);
    b_i = Cov_b * lambda / 2;

    u_i = inv(A) * b_i;

    % u_i are cols of U solution
    U(:, i) = u_i;

    % collect representer
    r_i = S * u_i;

    % r_i are cols of R
    R(:,i) = r_i;
end

I = eye(pt_size);
d = inv( I + R ) * S * Sol;

Sol2 = ( U * d );
```

```
% figure(2);
% for i = 1:size(elem,1)
%     patch( node(elem(i,2:4),1), node( elem(i,2:4),2), Sol2(elem(i,2:4)), 'FaceColor', 'inter
p' );
%     hold on;
% end
% colorbar;
% title('Potential from Measurement Data');
```

Error

```
bias = abs(Sol2 - Sol);

% figure(3);
% for i = 1:size(elem,1)
%     patch( node(elem(i,2:4),1), node( elem(i,2:4),2), bias(elem(i,2:4)), 'FaceColor', 'inter
p' );
%     hold on;
% end
% colorbar;
% title('Error');


rms_bias   = sqrt( sum( bias.^2 ) / (size(node,1)-3) );
rms_misfit = sqrt( sum( bias(elem(pt_elem,2)).^2 + bias(elem(pt_elem,3)).^2 + bias(elem(pt_ele
m,4)).^2 ) / 36 );
rms_dir    = sqrt( sum( bias(dnd).^2 ) / size(dnd,1) );
```

Inverse Noise

```
% uncorrelated noise with variance 0.05
Cov_d = (0.05)^2 * eye(12);
Cov_u = U * inv( I + R ) * Cov_d * transpose( inv( I + R ) ) * transpose( U );

inv_noise = sqrt( diag( Cov_u ) );
% figure(4);
% for i = 1:size(elem,1)
%     patch( node(elem(i,2:4),1), node( elem(i,2:4),2), inv_noise(elem(i,2:4)), 'FaceColor', '
interp' );
%     hold on;
% end
% colorbar;
% title('Inverse Noise');
```

Published with MATLAB® R2016a