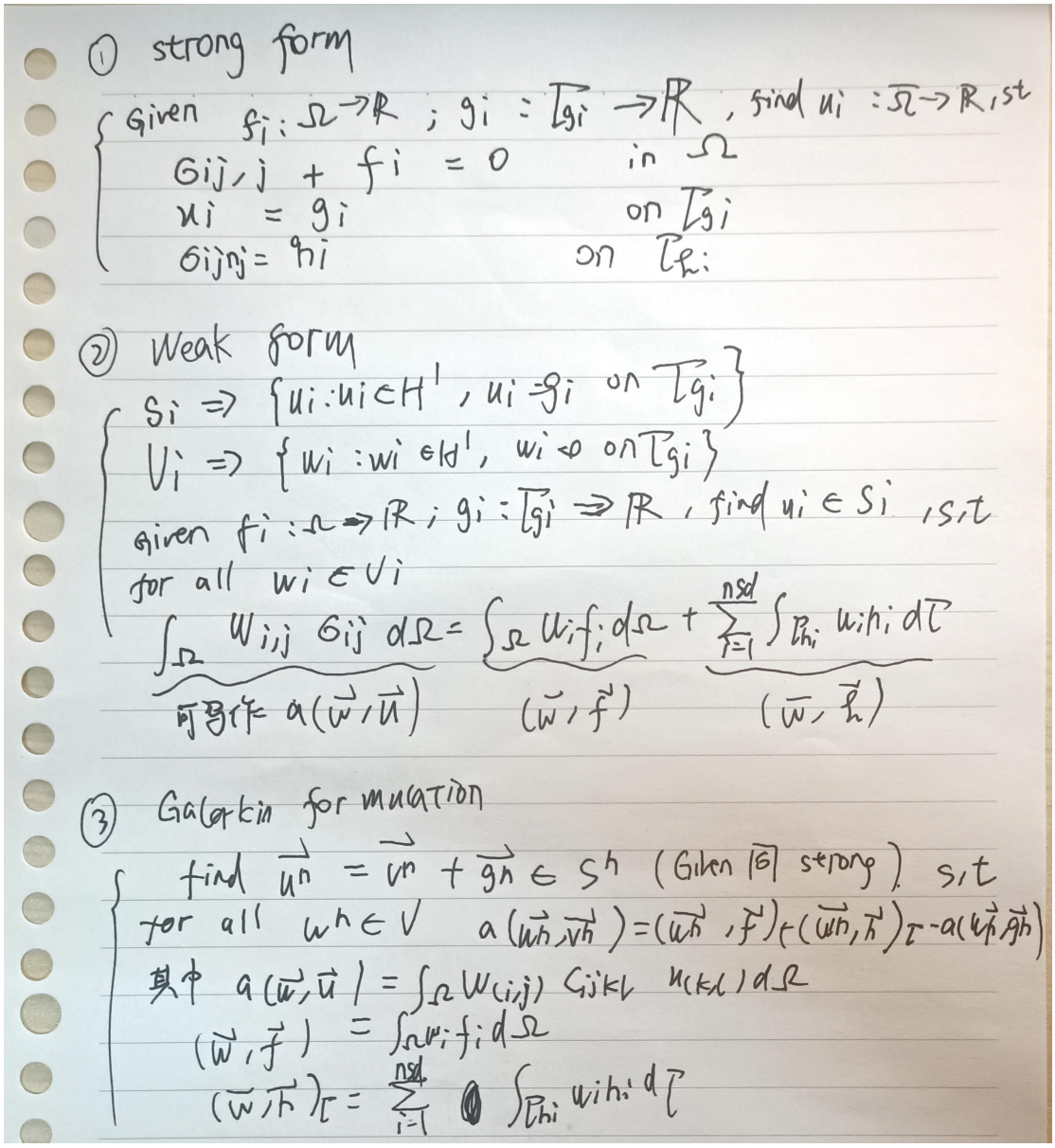
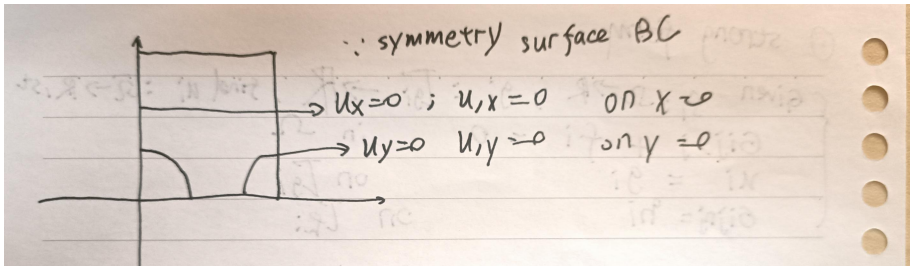


Boundary Conditions:



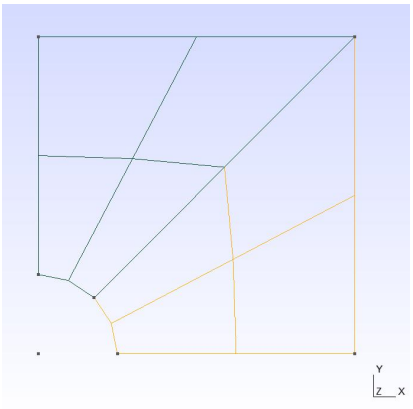
你可以在程序中使用依次（左，右，上，下）输入的方式来为四条边设置 Dirichlet 边界条件，或通过输入 g（已注释掉）的方式确立 Neumann 边界条件。

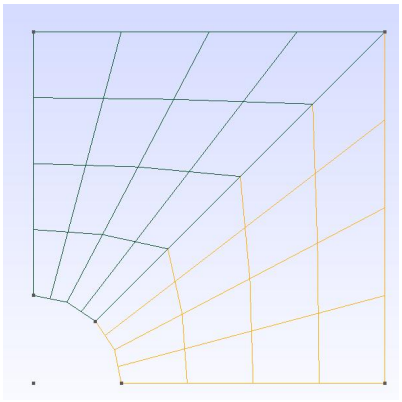
Implementation of the element stiffness matrix：

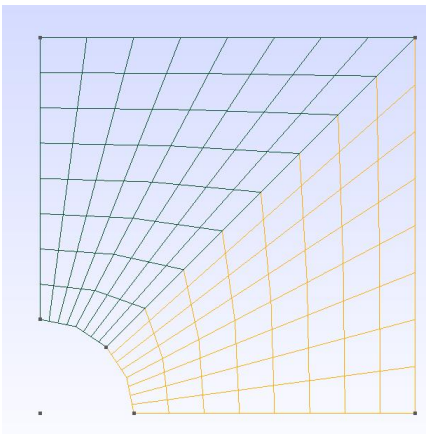
我在每一个节点上使用$B_a^T D B_b$的执行方法，Ke 可以通过以下公式来计算：

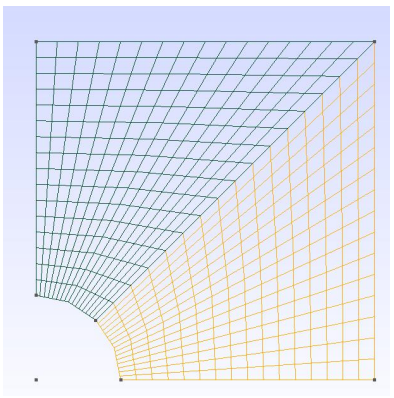
$$K_{ab}^{e} \cong \sum_{l=1}^{n_{int}} (B_a^T \tilde{D} B_b)_l$$

对于两个问题需要的，定制 mesh 的要求，我将每个 mesh 依照 refine 次数的不同分为 4 份，可以满足不同稠密度 mesh 的需求，如下图：









本题的 exact solution 如下：

$$\sigma_{rr}(r,\theta) = \frac{T_x}{2}\left(1-\frac{R^2}{r^2}\right)+\frac{T_x}{2}\left(1-4\frac{R^2}{r^2}+3\frac{R^4}{r^4}\right)\cos(2\theta)$$

$$\sigma_{\theta\theta}(r,\theta) = \frac{T_x}{2}\left(1+\frac{R^2}{r^2}\right)-\frac{T_x}{2}\left(1+3\frac{R^4}{r^4}\right)\cos(2\theta)$$

$$\sigma_{r\theta}(r,\theta) = -\frac{T_x}{2}\left(1+2\frac{R^2}{r^2}-3\frac{R^4}{r^4}\right)\sin(2\theta)$$

我使用摩尔圆法将这个 exact solution 转化为正交坐标系的形式，公式为：

通过摩尔圆计算的笛卡尔坐标系应力分量的公式总结如下：

- 圆心：$\sigma_m = \frac{\sigma_r + \sigma_\theta}{2}$
- 半径：$R = \sqrt{\left(\frac{\sigma_\theta - \sigma_r}{2}\right)^2 + \sigma_{r\theta}^2}$

笛卡尔坐标系中的应力分量：

- σ_x：$\sigma_x = \sigma_m + R$
- σ_y：$\sigma_y = \sigma_m - R$
- σ_{xy}：$\sigma_{xy} = R$

代码如下：

```
sigma_rr=sigmarr(Tx,thet,R,r);
sigma_tt=sigmatt(Tx,thet,R,r);
sigma_rt=sigmart(Tx,thet,R,r);

g=0;
% g=input('please enter the Neumann BC g!');

sigma_m=(sigma_rr+sigma_tt).*0.5;
sigma_Rad=sqrt(((sigma_rr+sigma_tt).*0.5).^2+sigma_rt.^2);

fx=sigma_m+sigma_Rad;
fy=sigma_m-sigma_Rad;
fxy=sigma_Rad;
```

为了能让使用者在四条边上自定义狄里赫莱边界条件，我设置了一个输入：

```
% Boundry condition
% Dirichle=input('Where to apply the BC? Please input in the order of: left,right,Above,Below.0 for not apply,1 for apply.');%你可以自行选择在哪几条边上施加边界。
Dirichle=[1,0,0,1];
BC=zeros(n_np,2);
if Dirichle(1)==1
    for i=1:n_np
        if coor(i,1)==-1
            BC(i)=coor(i);
        end
    end
end
if Dirichle(2)==1
    for i=1:n_np
        if coor(i,1)==1
            BC(i)=coor(i);
        end
    end
end
if Dirichle(3)==1
    for i=1:n_np
        if coor(i,2)==1
            BC(i)=coor(i);
        end
    end
end
if Dirichle(4)==1
    for i=1:n_np
        if coor(i,2)==-1
            BC(i)=coor(i);
        end
    end
end
```

输入边界条件之后就可以通过已从 msh 文件中读取的坐标来确定哪些点在边界上，以便后续 ID array 的完成.

```
% ID array
ID = zeros(n_np,2);
counter = 0;
for i=1:n_np
    if ((coor(i,1)==-1)&&(Dirichle(1)==1))||((coor(i,1)==1)&&(Dirichle(2)==1))
        ID(1,i)=65535;
        elseif ((coor(i,2)==-1)&&(Dirichle(3)==1))||((coor(i,2)==1)&&(Dirichle(4)==1))
        ID(2,i)=65535;
    end %标记ID array中为0的点
end
count=0;
for i=1:n_np
    for j=1:2
        if ID(i,j) == 0
            count =count+1;
            ID(i,j)=count;
        else
            ID(i,j)=0;
        end
    end
end
n_eq=count;

%LM array
LM =zeros(n_el,n_en,2);
ID_row_1=ID(:,1); ID_row_2=ID(:,2);
LM(:,:,1)=ID_row_1(IEN);
LM(:,:,2)=ID_row_2(IEN);%两个自由度—LM array需要分成两份。
```

与传热问题不同的是，应力应变问题对于每个节点都有 x 和 y 方向上的两个自由度，因此 LM array 需要一个长度为 2 的第三维。这里也可以将 LM array 的纵向或横向长度增加一倍来存储，但我个人更喜欢加一维。

接下来就可以开始数字计算。程序主体与老师写的传热问题的程序相近，但因为加了一个自由度，需做一些“本地化”处理。

```
% loop over element to assembly the matrix and vector
for ee = 1 : n_el
  x_ele = coor( IEN(ee, 1:n_en),1 );
  y_ele = coor( IEN(ee, 1:n_en),2 );

  k_ele = zeros(n_en, n_en,2,2); % element stiffness matrix,每个元素因为有2个自由度的缘故被分割为2x2
  f_ele = zeros(n_en, 2);    % element load vector

  for ll = 1 : n_int
    x_l = 0.0; y_l = 0.0;
    dx_dxi = 0.0; dx_deta = 0.0;
    dy_dxi = 0.0; dy_deta = 0.0;
    for aa = 1 : n_en
      x_l = x_l + x_ele(aa) * Quad(aa, xi(ll), eta(ll));
      y_l = y_l + y_ele(aa) * Quad(aa, xi(ll), eta(ll));
      [Na_xi, Na_eta] = Quad_grad(aa, xi(ll), eta(ll));
      dx_dxi  = dx_dxi  + x_ele(aa) * Na_xi;
      dx_deta = dx_deta + x_ele(aa) * Na_eta;
      dy_dxi  = dy_dxi  + y_ele(aa) * Na_xi;
      dy_deta = dy_deta + y_ele(aa) * Na_eta;
    end

    detJ = dx_dxi * dy_deta - dx_deta * dy_dxi;

    for aa = 1 : n_en
      Na = Quad(aa, xi(ll), eta(ll));
      [Na_xi, Na_eta] = Quad_grad(aa, xi(ll), eta(ll));
      Na_x = (Na_xi * dy_deta - Na_eta * dy_dxi) / detJ;
      Na_y = (-Na_xi * dx_deta + Na_eta * dx_dxi) / detJ;

      f_ele(aa,1)=f_ele(aa,1)+Na*weight(ll)*detJ*fx(IEN(ee,aa));
      f_ele(aa,2)=f_ele(aa,2)+Na*weight(ll)*detJ*fy(IEN(ee,aa));
```

这里由外而内是 element loop，quadrature loop 和 nodes loop（应该是这个名字）。计算出 element stiffness matrix:k_ele 和 element load vector:f_ele。

$K_{ab}^{e} \cong \sum_{l=1}^{n_{int}} (B_a^T \tilde{D} B_b)_l$，计算出 B，然后计算出 k_ele。

```
B=zeros(3,2);%获取B矩阵
for bb = 1 : n_en
  Nb = Quad(bb, xi(ll), eta(ll));
  [Nb_xi, Nb_eta] = Quad_grad(bb, xi(ll), eta(ll));
  Nb_x = (Nb_xi * dy_deta - Nb_eta * dy_dxi) / detJ;
  Nb_y = (-Nb_xi * dx_deta + Nb_eta * dx_dxi) / detJ;
  B(1,2)=Nb_x;B(2,2)=Nb_y;B(3,2)=Nb_x;B(3,1)=Nb_y;
  k_ele(aa,bb,:,:)=B'*D*B;
```

计算出大 K 和大 F。两个自由度，所以程序大致是老师写的传热程序这一步的两倍。

```
  for aa = 1 : n_en
    PP_1=LM(ee,aa,1);
    PP_2=LM(ee,aa,2);
    if PP_1> 0
      F(PP_1)=F(PP_1)+f_ele(aa,1);
      for bb = 1 : n_en
        QQ_1= LM(ee, bb,1);
         QQ_2= LM(ee, bb,2);
        if QQ_1 > 0
          K(PP_1, QQ_1) = K(PP_1, QQ_1) + k_ele(aa,bb,1,1);
        elseif QQ_2>0
              K(PP_1, QQ_2) = K(PP_1, QQ_2) + k_ele(aa,bb,1,2);
          % modify F with the boundary data
          % disp = [d_temp; g];
        end
      end
    end
    if PP_2> 0
      F(PP_2)=F(PP_2)+f_ele(aa,2);
      for bb = 1 : n_en
        QQ_1= LM(ee, bb,1);
         QQ_2= LM(ee, bb,2);
        if QQ_1 > 0
          K(PP_2, QQ_1) = K(PP_2, QQ_1) + k_ele(aa,bb,2,1);
        elseif QQ_2>0
              K(PP_2, QQ_2) = K(PP_2, QQ_2) + k_ele(aa,bb,2,2);
          % modify F with the boundary data
          % disp = [d_temp; g];
        end
      end
    end
  end
end
```

如果是纽曼边界条件，需将 disp=[d_temp;g]这一句去注释。

解出 dn，然后将节点的位移存储。

```
% solve the stiffness matrix
dn = K \ F;

% insert dn back into the vector for all nodes
disp = zeros(n_np, 2);

for ii = 1 : n_np
    for jj=1:2
     index = ID(ii,jj);
     if index > 0
       disp(ii,jj) = dn(index);
     else
       % disp = [d_temp; g]; nothing because g=0!
     end
    end
end
```