等参单元有限元分析
Matlab 程序实现

指导老师： 陈水福

小组成员： 唐敬哲、冯一笑

学　　号： 11312045、11312047

联系方式： 15068175316、13732253879

日　　期： 2014 年 1 月 6 日

目 录

一、 概况

等参数单元的形函数一般建立在一个参考坐标系下，它既被用来进行单元内位移插值，也被用来表示单元内任意一点的坐标。经过形函数的坐标变换，可以把物理空间下任意的四边形或六面体，甚至是曲边的四边形或曲面的六面体，转换成参考坐标系下的正方形或立方体。这些曲边或曲面的单元可以更精确地描述求解区域，而且应用更高阶的位移插值函数，因此具有较高的精度。另外，等参数单元形函数选择方法的统一性，非常有利于程序实现。

本程序就利用了等参单元的特性，采用 Matlab 进行编程，计算平面应力和平面应变的等参问题。通过该程序我们可以得到相关问题的应力以及位移解。为了增加用户的操作性，我们还利用 Matlab 自带的 GUI 功能编写了用户操作界面，并将计算结果绘制成云图显示。

二、 核心计算程序流程

本程序实现了四结点、八结点等参单元的有限元分析计算，包括了除网格划分之外的前处理、核心计算以及绘制应力云图、形成结果文件等后处理步骤。这里将部分前处理与核心计算部分的程序实现进行简要说明。

1. 定义计算模型

（1） 数据输入

进行核心计算前，需要对网格划分完成后的模型的结点信息、单元信息、材料特性、边界条件、荷载信息进行定义。

结点信息 gNode：二维数组，行指标表示结点的整体编号，第一、二列表示结点 X、Y 坐标。

单元信息 gElement：二维数组，行指标表示单元编号，前四列分别为单元结点序号位置所对应的结点在整体结点中的编号（对于八结点单元则为前八列），最后一列表示材料编号。

材料信息 gMaterial：二维数组，行指标表示材料种类编号，前三列分别表示材料的弹性模量、泊松比以及密度。

边界条件 gBC：二维数组，行指标表示边界约束编号，第一列表示约束位置的结点的整体编号，第二列表示约束方向（1 为 X 向，2 为 Y 向），第三列表示位移约束值（固定约束即为 0）。

边界线荷载信息 gDF：二维数组，行指标表示线荷载序号，第一列表示线荷载所在的单元编号，第二列表示线荷载作用边的编号（矩形单元取底边为 1 号边，逆时针旋转），第三、四列表示线荷载在该边两结点上的取值（对于八结点单元，每边上则有三个荷载值，同样按照逆时针方向排列），最后一列表示荷载作用方向（1 为 X 向，2 为 Y 向）。

（2） 确定平面问题类型

选择平面应力或平面应变问题，选择不同的弹性矩阵进行计算。

（3） 确定单元内部积分使用的高斯积分点数

确定了高斯积分点数，在进行单元内部积分时根据高斯积分点的两方向指标，调用 Gaussint 函数，得到积分点坐标与积分权值。本程序可以选择使用两点高斯积分或三点高斯积分。

2. 计算单元刚度矩阵并集成

形成单元刚度矩阵通过 StiffnessMatrix 函数来实现：

（1） 按照单元编号，从 1 号单元开始循环；

（2） 各单元内的积分操作按照高斯积分点指标进行循环和累加，根据两个方向的高斯积分点指标调用 Gaussint 函数得到积分点坐标和积分权值；

（3） 根据问题类型调用 ElasticityMatrix 函数，确定弹性矩阵；

（4） 根据各积分点的坐标调用 StrainMatrix 函数，得到该积分点的应变矩阵。计算时需要用到形函数对整体坐标的导数，这里也通过调用 N_xy 函数来实现；

（5） 根据积分点位置的局部坐标以及所在单元的结点坐标调用 Jacobi 函数，得到该积分点的雅克比矩阵。计算式需要用到形函数局部坐标的导数，这里也通过调用 N_xieta 函数来实现；

（6） 积分循环的关键语句为：

$$K = K + w * B' DB * \det(J)$$

式中 w 为积分点权值大小，B 为应变矩阵，D 为弹性矩阵，J 为雅克比矩阵，K 为不断循环叠加的单元刚度矩阵；

单元内部循环完成后，得到单元刚度矩阵，调用 AssembleStiffnessMatrix 函

数，将单元刚度矩阵集成到整体刚度矩阵 gK 中。

3. 计算重力的等效结点力并集成

形成重力的等效结点力通过 EquivalentGravityForce 函数实现：

（1） 按照单元编号，从 1 号单元开始循环；

（2） 各单元内的积分操作按照高斯积分点指标进行循环和累加，根据两个方向的高斯积分点指标调用 Gaussint 函数得到积分点坐标和积分权值；

（3） 根据积分点位置的局部坐标以及所在单元的结点坐标调用 Jacobi 函数，得到该积分点的雅克比矩阵。计算式需要用到形函数局部坐标的导数，这里也通过调用 N_xieta 函数来实现；

（4） 调用 ShapeFunction 函数，计算单元形函数矩阵；

（5） 积分循环的关键语句为：

$$egf = egf + w * N' * gf * det(J)$$

式中 w 为积分点权值大小，N 为单元形函数矩阵，gf 为体力向量（对重力即为$[0; -\rho g]$），J 为雅克比矩阵，egf 为不断循环叠加的单元的重力等效结点力向量；

（6） 单元内部循环完成后，得到单元的重力等效结点力向量 egf，调用 AssembleLoadVector 函数，将单元等效结点力集成到整体等效结点力向量 gF 中。

4. 计算分布力的等效结点力并集成

形成分布力的等效结点力通过 EquivalentDistForce 函数实现：

（1） 按照荷载序号，从 1 号单元开始循环；

（2） 各单元内的积分操作按照高斯积分点指标进行循环和累加，根据两个方向

的高斯积分点指标调用 Gaussint 函数得到积分点坐标和积分权值；

（3） 根据荷载作用边序号，调用 A 函数得到对应边的边界线微分；

（4） 根据荷载作用边序号，将作用边结点荷载值整合至单元结点荷载向量 df 中；

（5） 调用 ShapeFunction 函数，计算单元形函数矩阵；

（6） 积分循环的关键语句为：

$$edf = edf + N' * N * df * A * w$$

式中 w 为积分点权值大小，N 为单元形函数矩阵，df 为单元结点荷载向量（对于未作用均分布力的边上结点的荷载即为 0），A 为对应边的边界线微分，edf 为不断循环叠加的单元的分布力的等效结点力向量；

（7） 单元内部循环完成后，得到单元的分布力的等效结点力向量 edf，调用 AssembleLoadVector 函数，将单元等效结点力集成到整体等效结点力向量 gF 中。

5. 处理边界条件

经过以上步骤，整体刚度矩阵 gK 以及整体结点荷载向量 gF 已经形成。通过引入边界条件，这里采用乘大数法，对 gK 和 gF 中与约束对应位置的元素进行处理，得到可用于计算的 gK 和 gF。

6. 求解结点应力

求解有限元基本方程：gDelta = gK \ gF，得到 gDelta 为结点位移向量。

7. 求解单元结点应力

单元结点应力计算通过 gElementStress 函数实现：

（1） 按照单元编号，从 1 号单元开始循环；

（2） 每个单元根据整体编号下的结点位移向量 gDelta 形成单元结点位移向量 Delta；

（3） 按照公式 sigma = D*B*delta 求解单元结点的三个应力，其中 D 为单元弹性矩阵，B 为单元应变矩阵；

（4） 将 sigma 矩阵写入三维数组 gElementStress 中，该数组第一维表示单元号，第二位表示单元结点号，第三维表示结点三个应力值（X 向正应力、Y 向正应力以及剪应力）。

8. 计算结点平均应力

结点平均应力采用绕单元平均方法计算：

（1） 按照单元编号，从 1 号单元开始循环；

（2） 寻找公用该结点的单元，提取这些单元中该结点的应力以及个单元面积；

（3） 该结点的应力取这些单元中该点应力计算值的平均值，并以单元面积进行加权，写入二维数组 gNodeStress 中，gNodeStress 的行指标表示结点整体编号，三列分别表示该结点的三个平均应力。

三、 主程序

1. GUI 界面及代码

该界面主要包括以下功能：

1) 选择单元类型、计算问题类型、高斯积分点数目的选择
2) 结果应力位移图输出选择

```
function varargout = FEM_503(varargin)    %程序 GUI 初始

    gui_Singleton = 1;
    gui_State = struct('gui_Name',          mfilename, ...
                       'gui_Singleton',    gui_Singleton, ...
                       'gui_OpeningFcn', @FEM_503_OpeningFcn, ...
                       'gui_OutputFcn',    @FEM_503_OutputFcn, ...
                       'gui_LayoutFcn',    [] , ...
                       'gui_Callback',     []);
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});

   end

function FEM_503_OpeningFcn(hObject, ~, handles, varargin)    %生成 GUI 界面
   handles.output = hObject;
   guidata(hObject, handles);
   axes1_CreateFcn;
   set(handles.axes1,'visible','off')

function varargout = FEM_503_OutputFcn(~, ~, handles)
varargout{1} = handles.output;


function Calculate_Callback(hObject,eventdata,handles)    %模型计算按钮，点击以后开始计算相应例题

global element_option fj
    ETC1_SelectionChangeFcn(hObject, eventdata, handles);
    SorE_SelectionChangeFcn(hObject,eventdata,handles);
    Gauss_SelectionChangeFcn(hObject,eventdata,handles);
```

```
fj=0;
switch element_option
    case 1
        FemModel ;
        handles.axes1;
        cla(handles.axes1);
        colorbar( 'hide' );
        DisplayModel ;
        SolveModel ;
        DisplayResults ;
    case 2
        FemModel8 ;
        handles.axes1;
        cla(handles.axes1);
        colorbar( 'hide' );
        DisplayModel ;
        SolveModel8 ;
        DisplayResults ;
end
```

function SorE_SelectionChangeFcn(hObject,eventdata,handles) %单选框，选择平面应力或者平面应变

```
global opt COP

COP=get(hObject,'String');
switch COP
    case 'Strain'
        opt=1;
    case 'Stress'
        opt=2;
end
```

function Gauss_SelectionChangeFcn(hObject, ~, ~) %高斯积分点数目选择，2 点或者 3 点

```
global  int
```

```
GOP=get(hObject,'String');

switch GOP
    case '2 点高斯积分'
        int=2;
    case '3 点高斯积分'
        int=3;
end
```

function ETC1_SelectionChangeFcn(hObject, eventdata, handles) %等参单元数目选择

```
global element_option
echoice=get(hObject,'String');

switch echoice
    case '4 结点单元'
        element_option=1;
    case '8 结点单元'
        element_option=2;
end
```

function Outcome_SelectionChangeFcn(hObject,eventdata,handles) %读取绘制结果图类型

```
global  iStress
caseStress=get(hObject,'String');
 switch caseStress
        case 'x 方向正应力'
            iStress=1;
        case 'y 方向正应力'
            iStress=2;
        case 'xy 方向切应力'
            iStress=3;
        case 'x 方向位移'
            iStress=4;
        case 'y 方向位移'
            iStress=5;
```

```
 end

function output_Callback(hObject,eventdata, handles)     %确认画结果图像

    global element_option fj
    Outcome_SelectionChangeFcn(hObject,eventdata,handles)
    handles.axes1;
    cla(handles.axes1);
    switch element_option
        case 1
            PlotStress;
        case 2
            PlotStress8;
    end
    if fj==0
       winopen('result.txt');
       fj=fj+1;
    end
function axes1_CreateFcn(~, ~, ~) %显示图片的区域
function ETC1_CreateFcn(hObject, eventdata, handles)

function EXIT_Callback(~, ~, handles) %退出程序按钮
    delete(handles.figure1);
    clc;
    clear all;
```

2. 核心计算程序及代码

1) 模型基本信息函数

function FemModel

```
function FemModel
%   定义用于四结点单元的有限元模型
%       该案例为一梯形截面重力坝，上部宽 4m，下部宽 8m，高 8m，直角边承受静水压力，体力为重力
%   说明：
%       该函数定义平面杆系的有限元模型数据：
%         gNode -------  结点定义
%         gElement ---  单元定义
%         gMaterial ---  材料定义
%         gBC ---------  约束条件
%         gDF ---------  分布力
    clear gNode gElement gMaterial gBC gK gF gDF gDelta gElementStress gNodeStress int opt iStress
    global gNode gElement gMaterial gBC gDF
    m=18;n=32;                  % 网格个数
    length1 = 4 ;           % 上部宽度(x 方向)
    length2 = 8 ;           % 下部宽度(x 方向)
    height = 8 ;            % 高度(y 方向)
    dy = height/n ;         % 矩形单元的高度

    % 结点坐标
    gNode = zeros( (m+1)*(n+1), 2 ) ;
    for i=1:n+1
        for j=1:m+1
            k = (i-1)*(m+1)+j ;             % 结点号
            dx = (length2-(i-1)*(length2-length1)/n)/m ;
            xk = (j-1)*dx ;                 % 结点的 x 坐标
            yk = (i-1)*dy ;                 % 结点的 y 坐标
            gNode(k,:) = [xk, yk] ;
        end
    end
```

```
% 单元定义
gElement = zeros( m*n, 5 ) ;
for i=1:n
    for j=1:m
        k = (i-1)*m+j ;                 % 单元号
        n1 = (i-1)*(m+1)+j ;            % 第一个结点号
        n2 = (i-1)*(m+1)+j+1 ;          % 第二个结点号
        n3 = i*(m+1)+j+1 ;              % 第三个结点号
        n4 = i*(m+1)+j ;                % 第四个结点号
        gElement(k,:) = [n1, n2, n3, n4,1] ;
    end
end

% 材料性质
%                弹性模量      泊松比     密度
gMaterial = [3.0e10,      0.167,    2500] ;    %   混凝土

% 第一类约束条件
gBC = zeros( 2*(m+1), 3 ) ;
for j=1:(m+1)
    gBC(j,:) = [j, 1, 0.0] ;            % 底端结点 x 向固定
end
for j=(m+2):2*(m+1)
    gBC(j,:) = [j-(m+1), 2, 0.0] ;   % 底端结点 y 向固定
end

% 分布载荷（线性分布的水压力）
gDF = zeros( n, 5 ) ;
for i=1:n
    k = (i-1)*m+1 ;
    gDF(i,:) = [ k, 4, 1e4*(height-i*height/n), 1e4*(height-(i-1)*height/n), 1] ;
end

return
```

function FemModel8

```
function FemModel8
% 定义用于八结点等参单元的有限元模型
%       该案例为一段固支的变矩形截面悬臂梁，上部承受均布荷载
% 该函数定义平面杆系的有限元模型数据：
%           gNode -------  结点定义
%           gElement ----  单元定义
%           gMaterial ---  材料定义
%           gBC ---------  约束条件
%           gDF ---------  分布力
clear gNode gElement gMaterial gBC gK gF gDF gDelta gElementStress gNodeStress int opt iStress
    global gNode gElement gMaterial gBC gDF

    % 结点坐标
    node_number = 28 ;
    gNode = zeros( node_number, 2 ) ;
    j = 0 ;
    for i = 1:5:26        % 结点 1,6,11,16,21,26 的坐标
        gNode(i,1) = 0.2*j ;
        gNode(i,2) = 0 ;
        j = j + 1 ;
    end
    j = 0 ;
    for i = 4:5:24        % 结点 4,9,14,19,24 的坐标
        gNode(i,1) = 0.2*j+0.1 ;
        gNode(i,2) = 0 ;
        j = j+1 ;
    end
    j = 0 ;
    for i=2:5:27          % 结点 2,7,12,17,22,27 的坐标
        gNode(i,1) = 0.2*j ;
        gNode(i,2) = -0.1+0.01*j ;
        j = j+1 ;
    end
    j = 0 ;
```

```
for i=3:5:28          % 结点 3,8,13,18,23,28 的坐标
    gNode(i,1) = 0.2*j ;
    gNode(i,2) = -0.2+0.02*j ;
    j = j+1 ;
end
j = 0 ;
for i=5:5:25          % 结点 5,10,15,20,25 的坐标
    gNode(i,1) = 0.1+0.2*j ;
    gNode(i,2) = -0.19+0.02*j ;
    j = j+1 ;
end

% 单元定义
element_number = 5 ;
gElement = zeros( element_number, 9 ) ;
gElement(1,:)=[3   8   6   1    5   7   4   2   1] ;
for i=2:5
    gElement(i,1:8) = gElement(i-1,1:8)+5 ;
    gElement(i,9) = 1 ;
end

% 材料性质（弹性模量，泊松比，密度）
gMaterial = [2.0e11 0.3 7850] ;      % 钢材

% 约束条件（位移固定）
gBC = [1,1,0;1,2,0;2,1,0;2,2,0;3,1,0;3,2,0] ;

% 分布载荷（均布）
gDF                                                                                    =
[1,3,-1e6,-1e6,-1e6,2;2,3,-1e6,-1e6,-1e6,2;3,3,-1e6,-1e6,-1e6,2;4,3,-1e6,-1e6,-1e6,2;5,3,-1e6,-1e6,-1e6,2;] ;
return
```

function DisplayBC(color)

```
function DisplayBC( color )
```

```
%   用图形方式显示有限元模型的边界条件
%   输入参数：
%       color   ----   边界条件的颜色
%   返回值：
%       无
    global gNode gBC

    % 确定边界条件的大小
    xmin = min( gNode(:,1) ) ;
    xmax = max( gNode(:,1) ) ;
    factor = ( xmax - xmin ) / 25 ;

    [bc1_number,dummy] = size( gBC ) ;
    dBCSize = factor ;
    for i=1:bc1_number
        if( gBC( i, 2 ) == 1 )   %   x 方向约束
            x0 = gNode( gBC( i, 1 ), 1 ) ;
            y0 = gNode( gBC( i, 1 ), 2 ) ;
            x1 = x0 - dBCSize ;
            y1 = y0 + dBCSize/2 ;
            x2 = x1 ;
            y2 = y0 - dBCSize/2 ;
            hLine = line( [x0 x1 x2 x0], [y0 y1 y2 y0] ) ;
            set( hLine, 'Color', color ) ;

            xCenter = x1 - dBCSize/6 ;
            yCenter = y0 + dBCSize/4 ;
            radius = dBCSize/6 ;
            theta=0:pi/6:2*pi ;
            x = radius * cos( theta ) ;
            y = radius * sin( theta ) ;
            hLine = line( x+xCenter, y+yCenter ) ;
            set( hLine, 'Color', color ) ;

            hLine = line( x+xCenter, y+yCenter-dBCSize/2 ) ;
            set( hLine, 'Color', color ) ;
```

```
        x0 = x0 - dBCSize - dBCSize/3 ;
        y0 = y0 + dBCSize/2 ;
        x1 = x0 ;
        y1 = y0 - dBCSize ;
        hLine = line( [x0, x1], [y0, y1] ) ;
        set( hLine, 'Color', color ) ;

        x = [x0 x0-dBCSize/6] ;
        y = [y0 y0-dBCSize/6] ;
        hLine = line( x, y ) ;
        set( hLine, 'Color', color ) ;
        for j=1:1:4
            hLine = line( x, y - dBCSize/4*j );
            set( hLine, 'Color', color ) ;
        end
    else                                %   y 方向约束
        x0 = gNode( gBC( i, 1 ), 1 ) ;
        y0 = gNode( gBC( i, 1 ), 2 ) ;
        x1 = x0 - dBCSize/2 ;
        y1 = y0 - dBCSize ;
        x2 = x1 + dBCSize ;
        y2 = y1 ;
        hLine = line( [x0 x1 x2 x0], [y0 y1 y2 y0] ) ;
        set( hLine, 'Color', color ) ;

        xCenter = x0 - dBCSize/4 ;
        yCenter = y1 - dBCSize/6 ;
        radius = dBCSize/6 ;
        theta=0:pi/6:2*pi ;
        x = radius * cos( theta ) ;
        y = radius * sin( theta ) ;
        hLine = line( x+xCenter, y+yCenter ) ;
        set( hLine, 'Color', color ) ;

        hLine = line( x+xCenter+dBCSize/2, y+yCenter ) ;
```

```
            set( hLine, 'Color', color ) ;

            hLine = line( [x1, x1+dBCSize], [y1-dBCSize/3, y1-dBCSize/3] ) ;
            set( hLine, 'Color', color ) ;

            x = [x1 x1-dBCSize/6] ;
            y = [y1-dBCSize/3 y1-dBCSize/2] ;
            hLine = line( x, y ) ;
            set( hLine, 'Color', color ) ;
            for j=1:1:4
                hLine = line( x+dBCSize/4*j, y );
                set( hLine, 'Color', color ) ;
            end
        end
    end
return
```

function DisplayModel

```
function DisplayModel
%   用图形方式显示有限元模型
%   输入参数：
%       无
%   返回值：
%       无
    global gNode gElement gMaterial gBC

    figure ;
    axis equal ;
    axis off ;
    set( gcf, 'NumberTitle', 'off' ) ;
    set( gcf, 'Name', '有限元模型' ) ;

    % 根据不同的材料，显示单元颜色
    element_number = size( gElement,1 ) ;
    material_color = [ 'r','g','b','c','m','y','w','k'] ;
    for i=1:element_number
        x = gNode( gElement( i, 1:4 ), 1 ) ;
        y = gNode( gElement( i, 1:4 ), 2 ) ;
        color_index = mod( gElement( i, 5 ), length( material_color ) ) ;
        if color_index == 0
            color_index = length( material_color ) ;
        end
        patch( x, y, material_color( color_index ) ) ;
    end

    DisplayBC( 'blue' );

return
```

2) 力学计算函数

function SolveModel

```
%   求解四节点矩形等参单元有限元模型
%   输入参数：
%       无
%   返回值：
%       无
%   说明：
%         该函数求解有限元模型，过程如下
%           1. 计算单元刚度矩阵，集成整体刚度矩阵
%           2. 计算单元的等效结点力，集成整体结点力向量
%           3. 处理约束条件，修改整体刚度矩阵和结点力向量
%           4. 求解方程组，得到整体结点位移向量

    global gNode gElement gMaterial gBC gK gF gDF gDelta gElementStress gNodeStress

    % step1. 定义整体刚度矩阵和结点力向量
    node_number = size( gNode,1 );
    gK = sparse( node_number * 2, node_number * 2 ) ;
    gF = zeros(node_number * 2,1);

    % step2. 计算单元刚度矩阵，并集成到整体刚度矩阵中
    element_number = size( gElement,1 ) ;
    hbar=waitbar(0,'计算单元刚度矩阵并集成');
    for ie=1:element_number
        k = StiffnessMatrix( ie ) ;
        AssembleStiffnessMatrix( ie, k )
        waitbar(ie/element_number);
    end
    delete(hbar);

    % step3. 计算重力与边界分布力的等效结点力，并集成到整体结点力向量中
    hbar=waitbar(0,'计算重力的等效结点力向量并集成');
    for ie=1:element_number
```

```
    egf = EquivalentGravityForce( ie ) ;
    AssembleLoadVector( ie, egf ) ;
    waitbar(ie/element_number);
end
delete(hbar);
gdf_number=size(gDF,1);
hbar=waitbar(0,'计算分布力的等效结点力向量并集成');
for i=1:gdf_number
    edf = EquivalentDistForce( gDF(i,1),gDF(i,2),gDF(i,3),gDF(i,4),gDF(i,5) );
    AssembleLoadVector( gDF(i,1), edf ) ;
    waitbar(i/gdf_number);
end
delete(hbar);

% step4. 处理约束条件，修改刚度矩阵和结点力向量，采用乘大数法
bc1_number = size( gBC,1 ) ;
for ibc=1:bc1_number
    n = gBC(ibc, 1 ) ;
    d = gBC(ibc, 2 ) ;
    m = (n-1)*2 + d ;
    gF(m) = gBC(ibc, 3)* gK(m,m) * 1e15 ;
    gK(m,m) = gK(m,m) * 1e15 ;
end

% step5. 求解方程组，得到结点位移向量
gDelta = gK \ gF ;

% step6. 计算每个单元的结点应力，应力采用每结点周边单元的平均值
gElementStress = zeros( element_number, 4, 3) ;
delta = zeros( 8,1 ) ;
for ie = 1:element_number
    xi  = [ -1   1   1  -1] ;
    eta = [ -1  -1   1   1] ;
    for n=1:4
        B = StrainMatrix( ie, xi(n), eta(n) ) ;
        D = ElasticityMatrix( ie ) ;
```

```
            delta(1:2:7) = gDelta( gElement(ie,1:4)*2-1) ;
            delta(2:2:8) = gDelta( gElement(ie,1:4)*2) ;
            sigma = D*B*delta ;
            gElementStress( ie, n, :) = sigma ;
        end
    end
    gNodeStress=zeros(node_number,3);
    for i=1:node_number
        S=zeros(1,3);
        A=0;
        for ie=1:element_number
            x=gNode(gElement(ie,1:4),1);
            y=gNode(gElement(ie,1:4),2);
            A1=0.5*det([x(1),y(1),1;x(2),y(2),1;x(4),y(4),1]);
            A2=0.5*det([x(2),y(2),1;x(3),y(3),1;x(4),y(4),1]);
            area=A1+A2;
            for k=1:4
                if i==gElement(ie,k)
                    S(1)=S(1)+gElementStress(ie,k,1 )*area;
                    S(2)=S(2)+gElementStress(ie,k,2 )*area;
                    S(3)=S(3)+gElementStress(ie,k,3 )*area;
                    A=A+area;
                    break;
                end
            end
        end
        gNodeStress(i,1:3)=S/A;
    end
return

function AssembleLoadVector( ie, ef )
%   把单元的等效结点向量集成到整体结点力向量中
%   输入参数:
%       ie   --- 单元号
%       ef --- 单元的等效结点向量
%   返回值:
```

```
%       无
    global gElement gF

    for i=1:4
        for j=1:2
            m=(i-1)*2+j;
            M=(gElement(ie,i)-1)*2+j;
            gF(M)=gF(M)+ef(m);
        end
    end
return

function AssembleStiffnessMatrix( ie, k )
%   把单元刚度矩阵集成到整体刚度矩阵
%   输入参数:
%       ie   --- 单元号
%       k    --- 单元刚度矩阵
%   返回值:
%       无
    global gElement gK

    for i=1:4
        for j=1:4
            p=2*gElement(ie,i)-1;
            q=2*gElement(ie,j)-1;
            gK(p:p+1,q:q+1)=gK(p:p+1,q:q+1)+k(2*i-1:2*i,2*j-1:2*j);
        end
    end
return

function D = ElasticityMatrix( ie )
%   计算单元的弹性矩阵 D
%   输入参数：
%       ie ---------  单元号
%   返回值：
%       D   ---------  弹性矩阵 D
```

```
    global gElement gMaterial opt

    E=gMaterial(gElement(ie,5),1);     % 弹性模量
    mu=gMaterial(gElement(ie,5),2);     % 泊松比
    if opt == 1     % 平面应力的弹性常数
        A1=mu;
        A2=(1-mu)/2;
        A3=E/(1-mu^2);
    else                % 平面应变的弹性常数
        A1=mu/(1-mu);
        A2=(1-2*mu)/2/(1-mu);
        A3=E*(1-mu)/(1+mu)/(1-2*mu);
    end
    D=A3*[ 1    A1     0
           A1     1     0
            0     0    A2];
return

function egf = EquivalentGravityForce( ie )
%    计算重力的等效结点力
%    输入参数:
%        ie ----------    单元号
%    返回值:
%        egf ---------    重力的等效结点力向量
    global gElement gMaterial int

    egf=zeros(8,1);
    ro=gMaterial(gElement(ie,5),3);
    for i=1:int
        for j=1:int
            [x,wx]=GaussInt(i);
            [y,wy]=GaussInt(j);
            J=Jacobi(ie,x,y);
            N=ShapeFunction(x,y);
            egf=egf+N'*[0;-ro*9.8]*det(J)*wx*wy;
        end
```

```
    end
return

function edf = EquivalentDistForce( ie,iedge,p1,p2,idof )
%   计算分布荷载的等效结点力
%   输入参数:
%       ie      ----------  单元号
%       iedge ----------  施加分布荷载的单元的边代号，以下边为 1 号，逆时针旋转编号
%       p1,p2 ----------  对应边结点处分布荷载值大小，顺序按逆时针
%       idof   ----------  荷载作用方向，1 为 x 方向,2 为 y 方向
%   返回值:
%       edf     ----------  分布力的等效结点力向量
    global int

    edf=zeros(8,1);
    df=zeros(8,1);
    index=[idof,2+idof;2+idof,4+idof;4+idof,6+idof;6+idof,idof];
    df(index(iedge,1))=p1;
    df(index(iedge,2))=p2;

    if iedge==1
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(x,-1);
            edf=edf+N'*N*df*A(ie,iedge,x,-1)*w;
        end
    elseif iedge==2
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(1,x);
            edf=edf+N'*N*df*A(ie,iedge,1,x)*w;
        end
    elseif iedge==3
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(x,1);
```

```
            edf=edf+N'*N*df*A(ie,iedge,x,1)*w;
        end
    else
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(-1,x);
            edf=edf+N'*N*df*A(ie,iedge,-1,x)*w;
        end
    end
return

function [D,W] = GaussInt( i )
%   计算各高斯积分点的坐标和积分权值
%   输入参数：
%       i   -- 积分点指标
%   返回值：
%       D   -- 积分点坐标
%       W   -- 积分点权值
    global int

    GXY=[0.0,-0.577350269189626,-0.774596669241483;
         0.0, 0.577350269189626,                0.0;
         0.0,                0.0, 0.774596669241483];
    WXY=[2.0,                1.0, 0.555555555555556;
         0.0,                1.0, 0.888888888888889;
         0.0,                0.0, 0.555555555555556];

    D=GXY(i,int);
    W=WXY(i,int);
return

function J = Jacobi( ie, xi, eta )
%   计算雅克比矩阵
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
```

```
%   返回值：
%       J    -------  在局部坐标(xi,eta)处的雅克比矩阵
    global gNode gElement

    x=gNode(gElement(ie,1:4),1);
    y=gNode(gElement(ie,1:4),2);
    [N_xi,N_eta]=N_xieta(xi,eta);
    x_xi=N_xi*x;
    x_eta=N_eta*x;
    y_xi=N_xi*y;
    y_eta=N_eta*y;
    J=[x_xi,y_xi;x_eta,y_eta];
return

function [N_xi, N_eta] = N_xieta( xi, eta )
%   计算形函数对局部坐标的导数
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
%   返回值：
%       N_xi     -------  在局部坐标处的形函数对 xi 坐标的导数
%       N_eta    -------  在局部坐标处的形函数对 eta 坐标的导数

    N_xi=zeros(1,4);
    N_eta=zeros(1,4);

    N_xi(1)=-(1-eta)/4;
    N_eta(1)=-(1-xi)/4;
    N_xi(2)=(1-eta)/4;
    N_eta(2)=-(1+xi)/4;
    N_xi(3)=(1+eta)/4;
    N_eta(3)=(1+xi)/4;
    N_xi(4)=-(1+eta)/4;
    N_eta(4)=(1-xi)/4;
return
```

```
function [N_x, N_y] = N_xy( ie, xi, eta )
%   计算形函数对整体坐标的导数
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
%   返回值：
%       N_x   -------  在局部坐标处的形函数对 x 坐标的导数
%       N_y   -------  在局部坐标处的形函数对 y 坐标的导数

    J=Jacobi(ie,xi,eta);
    [N_xi,N_eta]=N_xieta(xi,eta);
    A=J\[N_xi;N_eta];
    N_x=A(1,:);
    N_y=A(2,:);
return

function A = A( ie,iedge,xi,eta )
%    单元边界线积分
%    输入参数：
%        ie ---------  单元号
%        xi,eta -----  局部坐标
%        iedge ----------   施加分布荷载的单元边代号，以下边为 1 号，逆时针旋转
%    返回值:
%        A ----------   边界线积分值

    J = Jacobi( ie, xi, eta );
    if iedge==1||iedge==3
        A=sqrt(J(1,1)^2+J(1,2)^2);
    else
        A=sqrt(J(2,1)^2+J(2,2)^2);
    end

return

function N = ShapeFunction( xi, eta )
%    计算形函数的值
```

```
%     输入参数:
%          ie ----------    单元号
%          xi, eta -----    单元内局部坐标
%     返回值:
%          N -----------    形函数的值

      N1=(1-xi)*(1-eta)/4;
      N2=(1+xi)*(1-eta)/4;
      N3=(1+xi)*(1+eta)/4;
      N4=(1-xi)*(1+eta)/4;
      N = [ N1    0    N2    0    N3    0    N4    0
                0    N1    0    N2 0      N3    0    N4 ];
return

function K = StiffnessMatrix( ie )
%   计算平面应变等参数单元的刚度矩阵
%   输入参数：
%        ie --  单元号
%   返回值：
%        K    --  单元刚度矩阵
      global int

      K=zeros(8,8);
      D=ElasticityMatrix(ie);
      for i=1:int
           for j=1:int
                [x,wx]=GaussInt(i);
                [y,wy]=GaussInt(j);
                B=StrainMatrix(ie,x,y);
                J=Jacobi(ie,x,y);
                K=K+wx*wy*B'*D*B*det(J);
           end
      end
return

function B = StrainMatrix( ie, xi, eta )
```

```
%   计算单元的应变矩阵 B
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
%   返回值：
%       B   ---------  在局部坐标处的应变矩阵 B

    [N_x,N_y]=N_xy(ie,xi,eta);
    B=zeros(3,8);
    for i=1:4
        B(1:3,(2*i-1):2*i) = [N_x(i),0;0,N_y(i);N_y(i),N_x(i)];
    end
return
```

function SolveModel8

```
%   求解八节点矩形等参单元有限元模型
%   输入参数：
%       无
%   返回值：
%       无
%   说明：
%         该函数求解有限元模型，过程如下
%           1. 计算单元刚度矩阵，集成整体刚度矩阵
%           2. 计算单元的等效结点力，集成整体结点力向量
%           3. 处理约束条件，修改整体刚度矩阵和结点力向量
%           4. 求解方程组，得到整体结点位移向量

    global gNode gElement gMaterial gBC gK gF gDF gDelta gElementStress gNodeStress

    % step1. 定义整体刚度矩阵和结点力向量
    node_number = size( gNode,1 );
    gK = sparse( node_number * 2, node_number * 2 ) ;
    gF = zeros(node_number * 2,1);

    % step2. 计算单元刚度矩阵，并集成到整体刚度矩阵中
    element_number = size( gElement,1 ) ;
    hbar=waitbar(0,'计算单元刚度矩阵并集成');
    for ie=1:element_number
        k = StiffnessMatrix( ie ) ;
        AssembleStiffnessMatrix( ie, k ) ;
        waitbar(ie/element_number);
    end
    delete(hbar);

    % step3. 计算单元的等效结点力，包括重力荷载与边界分布荷载，并集成到整体结点力向量中
    hbar=waitbar(0,'计算重力的等效结点力向量并集成');
    for ie=1:element_number
        egf = EquivalentGravityForce( ie ) ;
        AssembleLoadVector( ie, egf ) ;
```

```
    waitbar(ie/element_number);
end
delete(hbar);
gdf_number=size(gDF,1);
hbar=waitbar(0,'计算分布力的等效结点力向量并集成');
for i=1:gdf_number
    edf = EquivalentDistForce( gDF(i,1),gDF(i,2),gDF(i,3),gDF(i,4),gDF(i,5),gDF(i,6) );
    AssembleLoadVector( gDF(i,1), edf ) ;
    waitbar(i/gdf_number);
end
delete(hbar);

% step4. 处理约束条件，修改刚度矩阵和结点力向量，采用乘大数法
bc1_number = size( gBC,1 ) ;
for ibc=1:bc1_number
    n = gBC(ibc, 1 ) ;
    d = gBC(ibc, 2 ) ;
    m = (n-1)*2 + d ;
    gF(m) = gBC(ibc, 3)* gK(m,m) * 1e15 ;
    gK(m,m) = gK(m,m) * 1e15 ;
end

% step5. 求解方程组，得到结点位移向量
gDelta = gK \ gF ;

% step6. 计算每个单元的结点应力，应力采用每结点周边单元的平均值
gElementStress = zeros( element_number, 8, 3) ;
delta = zeros( 16,1 ) ;
for ie = 1:element_number
    xi  = [ -1   1   1  -1   0   1   0  -1 ] ;
    eta = [ -1  -1   1   1  -1   0   1   0 ] ;
    for n=1:8
        B = StrainMatrix( ie, xi(n), eta(n) ) ;
        D = ElasticityMatrix( ie ) ;
        delta(1:2:15) = gDelta( gElement(ie,1:8)*2-1) ;
        delta(2:2:16) = gDelta( gElement(ie,1:8)*2) ;
```

```
            sigma = D*B*delta ;
            gElementStress( ie, n, :) = sigma ;
        end
    end
    gNodeStress=zeros(node_number,3);
    for i=1:node_number
        S=zeros(1,3);
        A=0;
        for ie=1:element_number
            x=gNode(gElement(ie,1:4),1);
            y=gNode(gElement(ie,1:4),2);
            A1=0.5*det([x(1),y(1),1;x(2),y(2),1;x(4),y(4),1]);
            A2=0.5*det([x(2),y(2),1;x(3),y(3),1;x(4),y(4),1]);
            area=A1+A2;
            for k=1:8
                if i==gElement(ie,k)
                    S(1)=S(1)+gElementStress(ie,k,1 )*area;
                    S(2)=S(2)+gElementStress(ie,k,2 )*area;
                    S(3)=S(3)+gElementStress(ie,k,3 )*area;
                    A=A+area;
                    break;
                end
            end
        end
        gNodeStress(i,1:3)=S/A;
    end
return

function AssembleLoadVector( ie, ef )
%   把单元的等效结点向量集成到整体结点力向量中
%   输入参数:
%       ie   --- 单元号
%       ef --- 单元的等效结点向量
%   返回值:
%       无
    global gElement gF
```

```
    for i=1:4
        for j=1:2
            m=(i-1)*2+j;
            M=(gElement(ie,i)-1)*2+j;
            gF(M)=gF(M)+ef(m);
        end
    end
return

function AssembleStiffnessMatrix( ie, k )
%   把单元刚度矩阵集成到整体刚度矩阵
%   输入参数:
%       ie   --- 单元号
%       k    --- 单元刚度矩阵
%   返回值:
%       无
    global gElement gK

    for i=1:8
        for j=1:8
            p=2*gElement(ie,i)-1;
            q=2*gElement(ie,j)-1;
            gK(p:p+1,q:q+1)=gK(p:p+1,q:q+1)+k(2*i-1:2*i,2*j-1:2*j);
        end
    end
return

function D = ElasticityMatrix( ie )
%   计算单元的弹性矩阵 D
%   输入参数：
%       ie --------- 单元号
%   返回值：
%       D   --------- 弹性矩阵 D
    global gElement gMaterial opt
```

```
    E=gMaterial(gElement(ie,9),1);     % 弹性模量
    mu=gMaterial(gElement(ie,9),2);    % 泊松比
    if opt == 1     % 平面应力的弹性常数
        A1=mu;
        A2=(1-mu)/2;
        A3=E/(1-mu^2);
    else                % 平面应变的弹性常数
        A1=mu/(1-mu);
        A2=(1-2*mu)/2/(1-mu);
        A3=E*(1-mu)/(1+mu)/(1-2*mu);
    end
    D=A3*[ 1    A1     0
           A1     1     0
            0     0    A2];
return

function egf = EquivalentGravityForce( ie )
%    计算重力的等效结点力
%    输入参数:
%        ie ----------   单元号
%    返回值:
%        egf ---------   重力的等效结点力向量
    global gElement gMaterial int

    egf=zeros(16,1);
    ro=gMaterial(gElement(ie,9),3);
    for i=1:int
        for j=1:int
            [x,wx]=GaussInt(i);
            [y,wy]=GaussInt(j);
            J=Jacobi(ie,x,y);
            N=ShapeFunction(x,y);
            egf=egf+N'*[0;-ro*9.8]*det(J)*wx*wy;
        end
    end
return
```

```
function edf = EquivalentDistForce( ie,iedge,p1,p2,p3,idof )
%    计算分布力的等效结点力
%    输入参数:
%        ie          ----------   单元号
%        iedge       ----------   施加分布荷载的单元的边代号，以下边为 1 号，逆时针旋转编号
%        p1,p2,p3 ----------   对应边结点处分布荷载值大小，顺序按逆时针
%        idof        ----------   荷载作用方向，1 为 x 方向,2 为 y 方向
%    返回值:
%        edf         ----------   分布力的等效结点力向量
    global int

    edf=zeros(16,1);
    df=zeros(16,1);
    index=[1,5,2;2,6,3;3,7,4;4,8,1];
    df(2*(index(iedge,1)-1)+idof)=p1;
    df(2*(index(iedge,2)-1)+idof)=p2;
    df(2*(index(iedge,3)-1)+idof)=p3;

    if iedge==1
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(x,-1);
            edf=edf+N'*N*df*A(ie,iedge,x,-1)*w;
        end
    elseif iedge==2
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(1,x);
            edf=edf+N'*N*df*A(ie,iedge,1,x)*w;
        end
    elseif iedge==3
        for i=1:int
            [x,w]=GaussInt(i);
            N=ShapeFunction(x,1);
            edf=edf+N'*N*df*A(ie,iedge,x,1)*w;
```

```
            end
        else
            for i=1:int
                [x,w]=GaussInt(i);
                N=ShapeFunction(-1,x);
                edf=edf+N'*N*df*A(ie,iedge,-1,x)*w;
            end
        end

return

function [D,W] = GaussInt( i )
%   计算各高斯积分点的坐标和积分权值
%   输入参数：
%       i   -- 积分点指标
%   返回值：
%       D   -- 积分点坐标
%       W   -- 积分点权值
    global int

    GXY=[0.0,-0.577350269189626,-0.774596669241483;
         0.0, 0.577350269189626,                0.0;
         0.0,                0.0, 0.774596669241483];
    WXY=[2.0,                1.0, 0.555555555555556;
         0.0,                1.0, 0.888888888888889;
         0.0,                0.0, 0.555555555555556];

    D=GXY(i,int);
    W=WXY(i,int);
return

function J = Jacobi( ie, xi, eta )
%   计算雅克比矩阵
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
```

```
%   返回值：
%       J     -------  在局部坐标(xi,eta)处的雅克比矩阵
    global gNode gElement

    x=gNode(gElement(ie,1:8),1);
    y=gNode(gElement(ie,1:8),2);
    [N_xi,N_eta]=N_xieta(xi,eta);
    x_xi=N_xi*x;
    x_eta=N_eta*x;
    y_xi=N_xi*y;
    y_eta=N_eta*y;
    J=[x_xi,y_xi;x_eta,y_eta];
return

function [N_xi, N_eta] = N_xieta( xi, eta )
%   计算形函数对局部坐标的导数
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
%   返回值：
%       N_xi     -------  在局部坐标处的形函数对 xi 坐标的导数
%       N_eta    -------  在局部坐标处的形函数对 eta 坐标的导数

    x = [ -1, 1, 1, -1 ] ;
    e = [ -1, -1, 1, 1 ] ;
    N_xi  = zeros( 1, 8 ) ;
    N_eta = zeros( 1, 8 ) ;

    N_xi( 5 )  = xi*(eta-1) ;
    N_eta( 5 ) = 0.5*(xi^2-1) ;
    N_xi( 6 )  = 0.5*(1-eta^2) ;
    N_eta( 6 ) = -eta*(xi+1) ;
    N_xi( 7 )  = -xi*(eta+1) ;
    N_eta( 7 ) = 0.5*(1-xi^2) ;
    N_xi( 8 )  = 0.5*(eta^2-1) ;
    N_eta( 8 ) = eta*(xi-1) ;
```

```
    N_xi(1)   = x(1)*(1+e(1)*eta)/4 - 0.5*( N_xi(5)   + N_xi(8) );
    N_eta(1) = e(1)*(1+x(1)*xi)/4   - 0.5*( N_eta(5) + N_eta(8) ) ;
    N_xi(2)   = x(2)*(1+e(2)*eta)/4 - 0.5*( N_xi(5)   + N_xi(6) );
    N_eta(2) = e(2)*(1+x(2)*xi)/4   - 0.5*( N_eta(5) + N_eta(6) ) ;
    N_xi(3)   = x(3)*(1+e(3)*eta)/4 - 0.5*( N_xi(6)   + N_xi(7) );
    N_eta(3) = e(3)*(1+x(3)*xi)/4   - 0.5*( N_eta(6) + N_eta(7) ) ;
    N_xi(4)   = x(4)*(1+e(4)*eta)/4 - 0.5*( N_xi(7)   + N_xi(8) );
    N_eta(4) = e(4)*(1+x(4)*xi)/4   - 0.5*( N_eta(7) + N_eta(8) ) ;
return

function [N_x, N_y] = N_xy( ie, xi, eta )
%   计算形函数对整体坐标的导数
%   输入参数：
%       ie --------- 单元号
%       xi,eta ----- 局部坐标
%   返回值：
%       N_x   ------- 在局部坐标处的形函数对 x 坐标的导数
%       N_y   ------- 在局部坐标处的形函数对 y 坐标的导数

    J=Jacobi(ie,xi,eta);
    [N_xi,N_eta]=N_xieta(xi,eta);
    A=J\[N_xi;N_eta];
    N_x=A(1,:);
    N_y=A(2,:);
return

function A = A( ie,iedge,xi,eta )
%    单元边界线积分
%    输入参数：
%        ie --------- 单元号
%        xi,eta ----- 局部坐标
%        iedge ----------   施加分布荷载的单元边代号，以下边为 1 号，逆时针旋转
%    返回值:
%        A -----------   边界线积分值
```

```
    J = Jacobi( ie, xi, eta );
    if iedge==1||iedge==3
        A=sqrt(J(1,1)^2+J(1,2)^2);
    else
        A=sqrt(J(2,1)^2+J(2,2)^2);
    end
return

function N = ShapeFunction( xi, eta )
%   计算形函数的值
%   输入参数:
%       ie ----------   单元号
%       xi, eta -----   单元内局部坐标
%   返回值:
%       N -----------   形函数的值

    N5=(eta-1)*(xi^2-1)/2 ;
    N6=(xi+1)*(1-eta^2)/2 ;
    N7=(eta+1)*(1-xi^2)/2 ;
    N8=(xi-1)*(eta^2-1)/2 ;
    N1=(1-xi)*(1-eta)/4-0.5*(N8+N5) ;
    N2=(1+xi)*(1-eta)/4-0.5*(N5+N6) ;
    N3=(1+xi)*(1+eta)/4-0.5*(N6+N7) ;
    N4=(1-xi)*(1+eta)/4-0.5*(N7+N8) ;
    N = [ N1   0   N2   0   N3   0   N4   0   N5   0   N6   0   N7   0   N8   0
          0   N1   0   N2 0    N3   0   N4   0   N5   0   N6 0    N7   0   N8];
return

function K = StiffnessMatrix( ie )
%   计算等参数单元的刚度矩阵
%   输入参数：
%       ie --  单元号
%   返回值：
%       K   --  单元刚度矩阵
    global int
```

```
    K=zeros(16,16);
    D=ElasticityMatrix(ie);
    for i=1:int
        for j=1:int
            [x,wx]=GaussInt(i);
            [y,wy]=GaussInt(j);
            B=StrainMatrix(ie,x,y);
            J=Jacobi(ie,x,y);
            K=K+wx*wy*B'*D*B*det(J);
        end
    end
return

function B = StrainMatrix( ie, xi, eta )
%   计算单元的应变矩阵 B
%   输入参数：
%       ie ---------  单元号
%       xi,eta -----  局部坐标
%   返回值：
%       B   ---------  在局部坐标处的应变矩阵 B

    [N_x,N_y]=N_xy(ie,xi,eta);
    B=zeros(3,16);
    for i=1:8
        B(1:3,(2*i-1):2*i) = [N_x(i),0;0,N_y(i);N_y(i),N_x(i)];
    end
return 绘制应力云图：PlotStress、PlotStress8
```

3) 结果输出及图形绘制函数

function PlotStress

```
function PlotStress
%   显示应力云图
%   输入参数：
%       iStress --- 应力分量指示，它可以是下面的值
%                       1   --   x 方向正应力
%                       2   --   y 方向正应力
%                       2   --   xy 方向切应力
%                       3   --   x 方向正位移
%                       4   --   y 方向正位移
%   返回值：无
    global gNode gElement gNodeStress gDelta iStress
    switch iStress
        case 1
            title='x 方向正应力';
        case 2
            title='y 方向正应力';
        case 3
            title='xy 方向切应力';
        case 4
            title='x 方向位移';
        case 5
            title='y 方向位移';
    end
    axis equal ;
    axis off ;
    set( gcf, 'NumberTitle', 'off' ) ;
    set( gcf, 'Name', title ) ;

    element_number = size( gElement,1 ) ;
    node_number = size( gNode,1 ) ;
    if iStress==1||iStress==2||iStress==3
        stressMin = min(min( gNodeStress( :, iStress ) ) );
```

```
        stressMax = max(max( gNodeStress( :, iStress ) ) );
        caxis( [stressMin, stressMax] ) ;
        colormap( 'jet' ) ;
        for ie=1:element_number
            x = gNode( gElement( ie, 1:4 ), 1 ) ;
            y = gNode( gElement( ie, 1:4 ), 2 ) ;
            c = gNodeStress( gElement( ie, 1:4 ) , iStress ) ;
            set( patch( x, y, c ), 'EdgeColor', 'interp' ) ;
        end
        yTick = stressMin:(stressMax-stressMin)/10:stressMax ;
        Label = cell( 1, length(yTick) );
        for i=1:length(yTick)
            Label{i} = sprintf( '%.2fMPa', yTick(i)/1e6 ) ;
        end
        set( colorbar( 'vert' ), 'YTick', yTick, 'YTickLabelMode', 'Manual', 'YTickLabel', Label ) ;
    else
        dispMin = min( gDelta( iStress-3:2:node_number ) ) ;
        dispMax = max( gDelta( iStress-3:2:node_number ) ) ;
        caxis( [dispMin, dispMax] ) ;
        colormap( 'jet' ) ;
        for ie=1:element_number
            x = gNode( gElement( ie, 1:4 ), 1 ) ;
            y = gNode( gElement( ie, 1:4 ), 2 ) ;
            c = gDelta( 2*(gElement( ie, 1:4 )-1)+iStress-3 ) ;
            set( patch( x, y, c ), 'EdgeColor', 'interp' ) ;
        end
        yTick = dispMin:(dispMax-dispMin)/10:dispMax ;
        Label = cell( 1, length(yTick) );
        for i=1:length(yTick)
            Label{i} = sprintf( '%.2fmm', yTick(i)/1e3 ) ;
        end
        set( colorbar( 'vert' ), 'YTick', yTick, 'YTickLabelMode', 'Manual', 'YTickLabel', Label ) ;
    end
Return
```

function PlotStress8

```
function PlotStress8
%   显示应力云图
%   输入参数：
%       iStress --- 应力分量指示，它可以是下面的值
%                       1   --   x 方向正应力
%                       2   --   y 方向正应力
%                       2   --   xy 方向切应力
%                       3   --   x 方向正位移
%                       4   --   y 方向正位移
%   返回值：无
    global gNode gElement gNodeStress gDelta iStress
    switch iStress
        case 1
            title='x 方向正应力';
        case 2
            title='y 方向正应力';
        case 3
            title='xy 方向切应力';
        case 4
            title='x 方向位移';
        case 5
            title='y 方向位移';
    end
     axis equal ;
    axis off ;
    set( gcf, 'NumberTitle', 'off' ) ;
    set( gcf, 'Name', title ) ;
       element_number = size( gElement,1 ) ;
    node_number = size( gNode,1 ) ;
    if iStress==1||iStress==2||iStress==3
        stressMin = min(min( gNodeStress( :, iStress )) );
        stressMax = max(max( gNodeStress( :, iStress ) ));
        caxis( [stressMin, stressMax] ) ;
        colormap( 'jet' ) ;
```

```
        for ie=1:element_number
            index=[1,5,2,6,3,7,4,8];
            x = gNode( gElement( ie, index(1:8) ), 1 ) ;
            y = gNode( gElement( ie, index(1:8) ), 2 ) ;
            c = gNodeStress( gElement( ie, index(1:8) ) , iStress ) ;
            set( patch( x, y, c ), 'EdgeColor', 'interp' ) ;
        end
        yTick = stressMin:(stressMax-stressMin)/10:stressMax ;
        Label = cell( 1, length(yTick) );
        for i=1:length(yTick)
            Label{i} = sprintf( '%.2fMPa', yTick(i)/1e6 ) ;
        end
        set( colorbar( 'vert' ), 'YTick', yTick, 'YTickLabelMode', 'Manual', 'YTickLabel', Label ) ;
    else
        dispMin = min( gDelta( iStress-3:2:node_number ) ) ;
        dispMax = max( gDelta( iStress-3:2:node_number ) ) ;
        caxis( [dispMin, dispMax] ) ;
        colormap( 'jet' ) ;
        for ie=1:element_number
            index=[1,5,2,6,3,7,4,8];
            x = gNode( gElement( ie, index(1:8) ), 1 ) ;
            y = gNode( gElement( ie, index(1:8) ), 2 ) ;
            c = gDelta( 2*(gElement( ie, index(1:8) )-1)+iStress-3 ) ;
            set( patch( x, y, c ), 'EdgeColor', 'interp' ) ;
        end
        yTick = dispMin:(dispMax-dispMin)/10:dispMax ;
        Label = cell( 1, length(yTick) );
        for i=1:length(yTick)
            Label{i} = sprintf( '%.2fmm', yTick(i)/1e3 ) ;
        end
        set( colorbar( 'vert' ), 'YTick', yTick, 'YTickLabelMode', 'Manual', 'YTickLabel', Label ) ;
    end
        % PlotStressContour( iStress, 10, 'white' ) ;
return
```

function DisplayResults

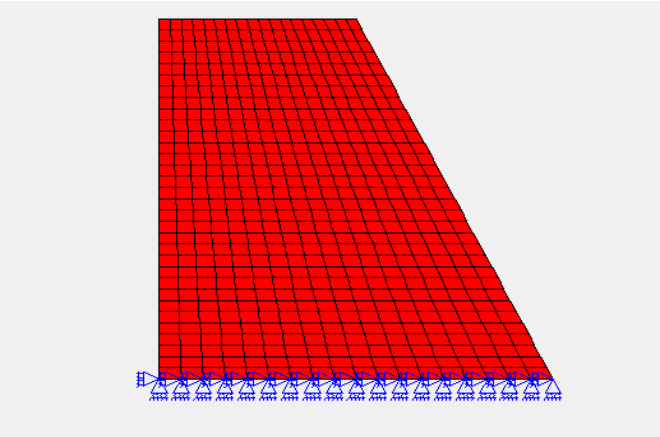
```
function DisplayResults
%   显示计算结果
%   输入参数：
%       无
%   返回值：
%       无

    global gNode gDelta gNodeStress

    fp=fopen('result.txt','wt');
    fprintf(fp, '结点位移   单位：m\n' ) ;
    fprintf(fp,'   结点号              x 方向位移                    y 方向位移\n' ) ;
    node_number = size( gNode,1 ) ;
    for i=1:node_number
        fprintf(fp,   '%6d          %16.8e          %16.8e\n',...
                    i, gDelta((i-1)*2+1), gDelta((i-1)*2+2) ) ;
    end
    fprintf(fp, '结点应力   单位：N/m^2\n' ) ;
    fprintf(fp, '   结点号              x 方向正应力                  y 方向正应力                  剪应力\n' ) ;
    node_number = size( gNode,1 ) ;
    for i=1:node_number
        fprintf(fp,   '%6d          %16.8e          %16.8e          %16.8e\n',...
                    i, gNodeStress(i,1), gNodeStress(i,2), gNodeStress(i,3) ) ;
    end
    fclose(fp);
return 算例及结果
```

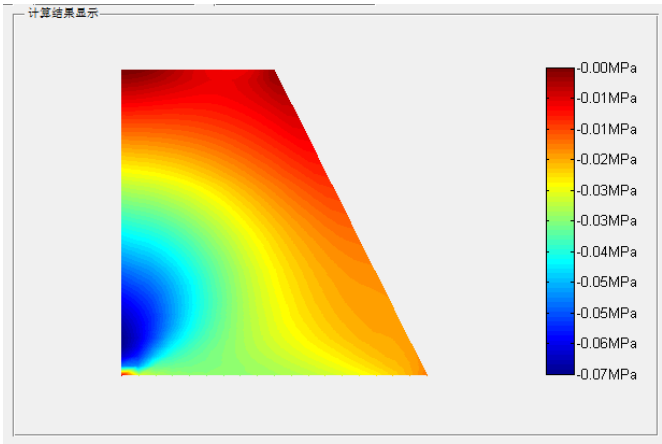
3. 算例 1

该案例为一梯形截面重力坝，上部宽 4m，下部宽 8m，高 8m，直角边承受静水压力，体力为重力，底端结点两方向固定。

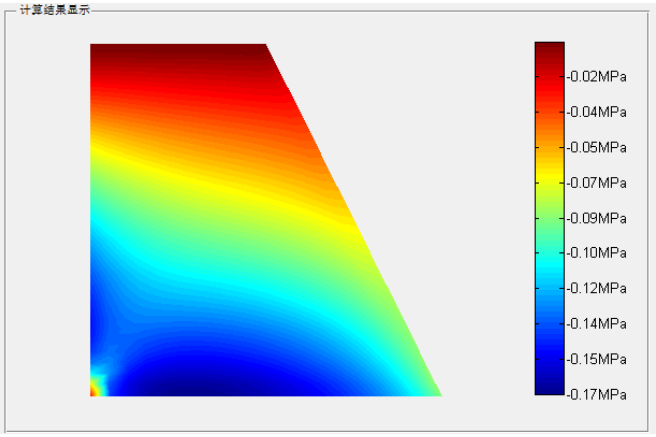


结构模型与网格划分图

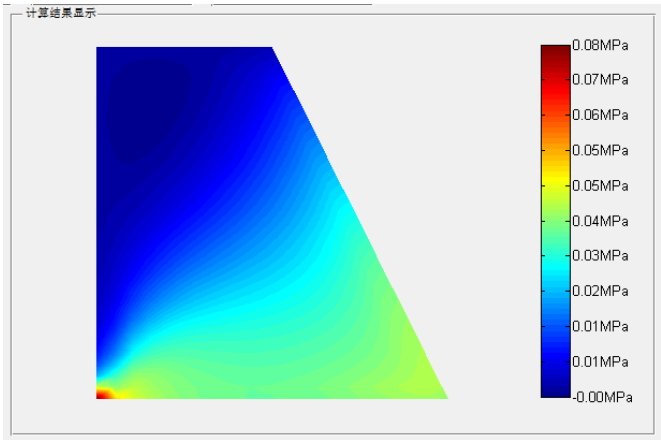
计算结果：



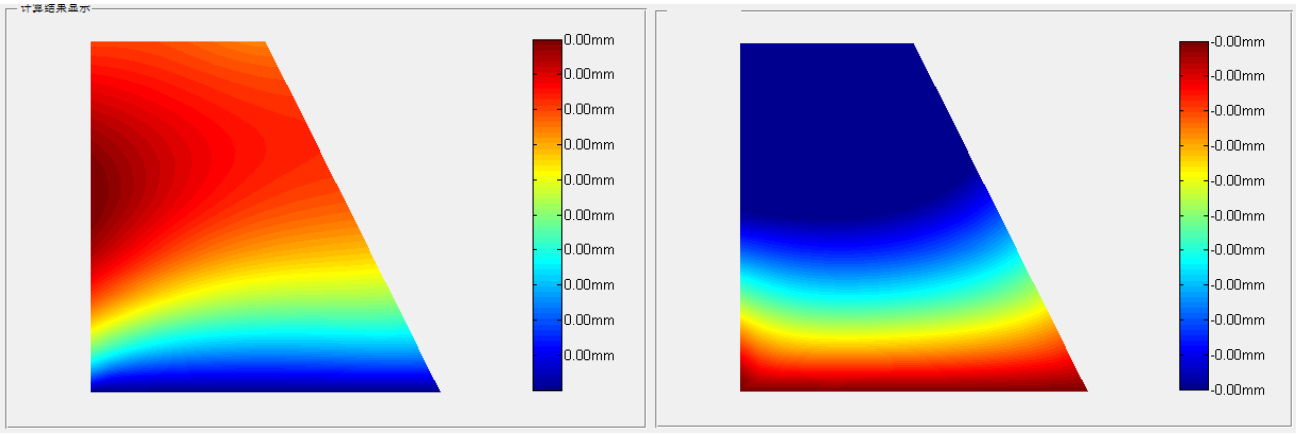
X 方向正应力



Y 方向正应力



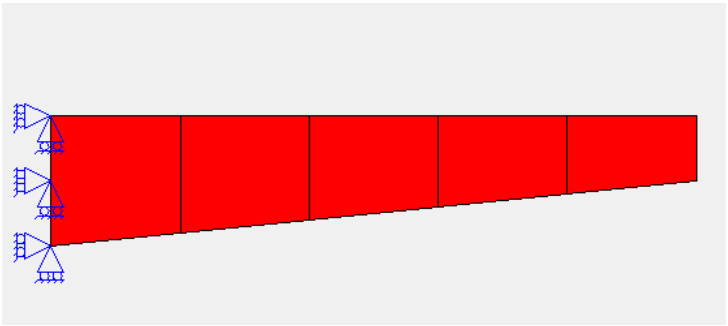
切应力



X 方向位移 **Y 方向位移**

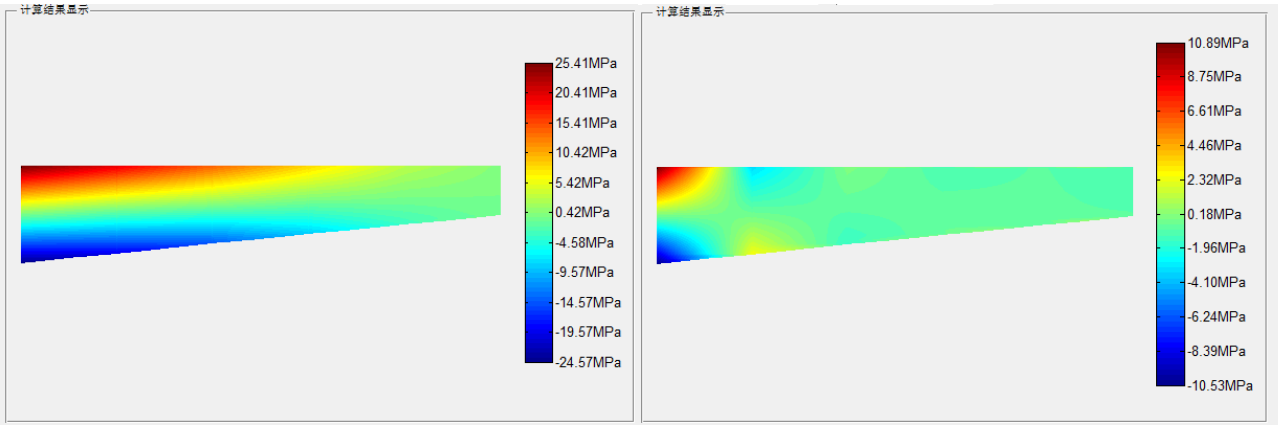
4. 算例 2

该案例为一端固支的变矩形截面悬臂梁，上部承受均布荷载，左端结点两方向固定。

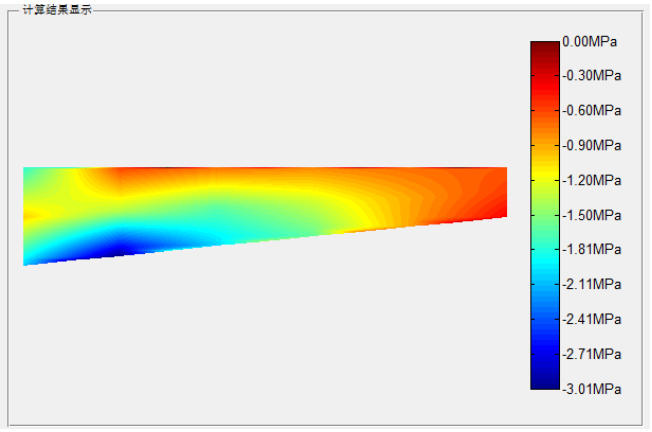


结构模型与网格划分图

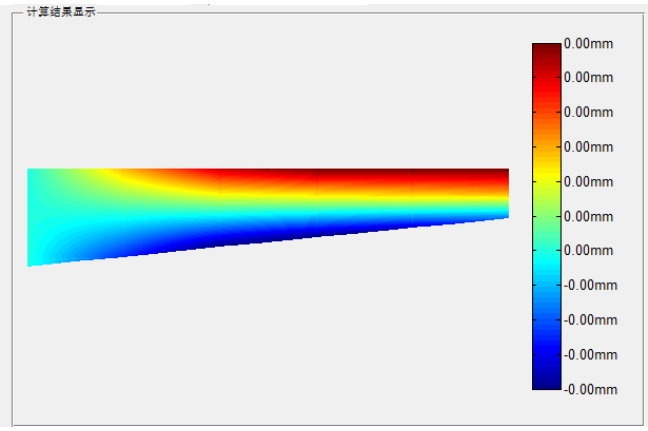
计算结果：



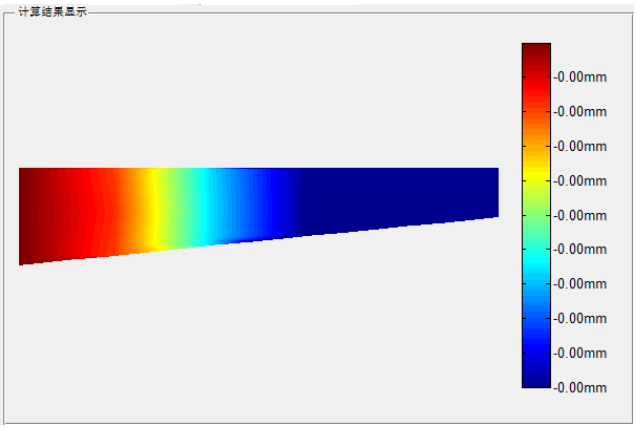
X 方向正应力 **Y 方向正应力**



切应力



X 方向位移



Y 方向位移

四、 感想与感谢

经过一个多月的学习与合作，我们小组完成了对该程序的编写。从中我们初步了解了等参单元是如何在有限元程序中工作的。

并且在程序计算结果得到专业有限元程序验证正确后，体会到了成功的喜悦。

此外我们还想感谢徐荣桥老师在编程方面给予的指导。