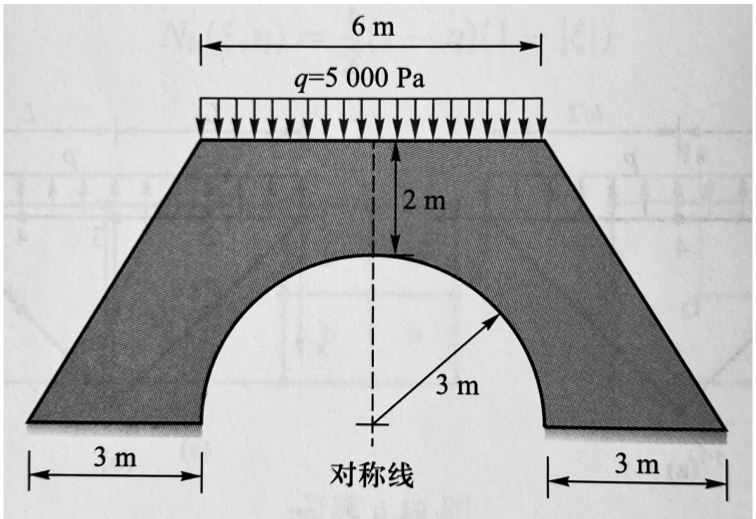
1 习题 4.11

一涵洞顶部路面受均布载荷 $q = 5000Pa$ 作用，尺寸如习题 4.11 图所示。将本问题简化为平面应变问题，取弹性模量 $E = 70Gpa$，$\nu = 0.3$。考虑到对称性，可只取涵洞的一半进行分析。请利用 elasticity2d-python 程序确定最大主应力的位置和大小。



习题 4.11 图

1.1 前处理

使用四节点四边形单元绘制网格，

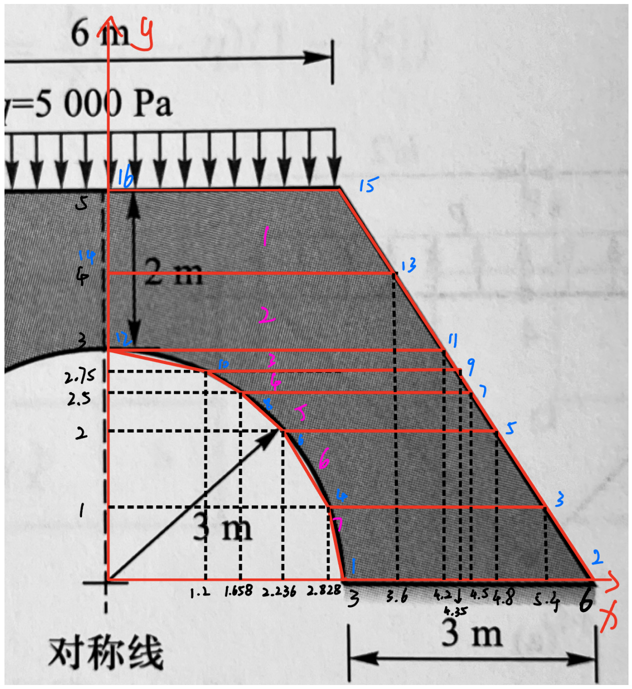


图 1: Mesh

将问题简化为平面应变问题时，单元的厚度视作 1，将所有数据组织为.json 文件，

```
 {
"Title": "Exercise 4-11 (7 element mesh)",

"nsd": 2,
"ndof": 2,
"nnp": 16,
"nel": 7,
"nen": 4,

"E": 70.0E+9,
"nu": 0.3,

"nd": 4,
"ngp": 2,

"flags": [2,2,2,2,0,0,0,0,0,0,
          0,0,0,0,0,0,0,0,0,0,
```

```
                0,0,0,0,0,0,0,0,0,0,
                0,0],

  "plane_strain": 0,
  "plot_mesh": "yes",
  "plot_nod": "yes",
  "plot_disp": "yes",
  "compute_stress": "yes",
  "plot_stress_xx": "yes",
  "plot_mises": "yes",
  "plot_tex": "no",
  "fact": 1e5,
  "print_disp":"no",

  "nbe": 1,
  "n_bc": [
    [16],
    [15],
    [0.0],
    [-5000.0],
    [0.0],
    [-5000.0]
  ],

  "x": [3, 6, 5.4, 2.828, 4.8, 2.236, 4.5, 1.658, 4.35, 1.2,
          4.2, 0, 3.6, 0, 3, 0],
  "y": [0, 0, 1, 1, 2, 2, 2.5, 2.5, 2.75, 2.75,
          3, 3, 4, 4, 5, 5],
  "IEN": [
    [1, 4, 6, 8, 10, 12, 14],
    [2, 3, 5, 7, 9, 11, 13],
    [3, 5, 7, 9, 11, 13, 15],
    [4, 6, 8, 10, 12, 14, 16]
  ]
}
```

1.2 有限元求解及应力计算

绘制好网格的模型如图 1 所示。运行程序，计算得到如下结果:

```
  Mesh Params
No. of Elements  7
No. of Nodes     16
No. of Equations 32

Condition number of stiffness matrix:  4406.285081770123

solution d
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
 -1.05105191e-07  2.43451008e-07 -1.67551770e-07 -4.92066905e-07
 -5.58272981e-07  2.74866405e-07 -5.99390739e-07 -1.20058869e-06
 -8.76722162e-07  1.87084832e-07 -9.25507108e-07 -1.74304686e-06
 -1.05289216e-06  1.11453892e-07 -1.10754182e-06 -2.12536319e-06
 -1.23496778e-06  1.02919016e-08 -1.30712662e-06 -3.02091620e-06
 -1.97386144e-06 -4.49776049e-07 -2.03779892e-06 -3.10904630e-06
 -2.72579606e-06 -9.45763159e-07 -2.78201092e-06 -3.18773105e-06]

reaction f =
 [[ 1377.66363906]
 [22500.        ]
 [-1377.66363906]
 [-7500.        ]]

                         Stress at Gauss Points
--------------------------------------------------------------------------------
Element  0
--------------
          x-coord                  y-coord                  s_xx
                          s_yy                    s_xy
          3.5785130092332094       0.21132486499999997      -7096.031828523167
                  -24711.74179536933       -2685.3943617597492
          5.258344194986792        0.21132486499999997      3008.2515642677836
                  8969.20284726717         -1684.547338177653
          3.4269891385467908       0.7886751350000001       -5025.980433203989
                  -21069.478484882682      1840.8928375259982
          4.96415365723321         0.7886751350000001       6016.097461884891
                   15737.44783208025        2934.630054293509
Element  1
```

```
--------------------
        x-coord                 y-coord                 s_xx                 s_yy                 s_xy
        3.246065967111462       1.211324865     -6254.024082501011      -26106.488288170618     -2045.2990269846907
        4.730034793808539       1.211324865     3641.490616676452       6878.560709087594       -2374.032069477466
        2.9032985395285382      1.788675135     -4330.317152994832      -18583.701899417094     2378.305844520683
        4.384600699551462       1.788675135     5583.011628943084       14460.727373709295      2048.981010918189
Element   2
--------------------
        x-coord                 y-coord                 s_xx                 s_yy                 s_xy
        2.6681061610917007      2.1056624325    -4988.388623930512      -20430.15915457943      -885.067443055659
        4.1823506074383 2.1056624325    2688.5982738171374      5159.797171246064       -787.5396772570106
        2.3683160590982997      2.3943375675    -3506.088693960885      -15641.00347755659      742.1241739592627
        3.975227172371701       2.3943375675    3728.1849871198874      8473.242126045983       834.0277500222601
Element   3
--------------------
        x-coord                 y-coord                 s_xx                 s_yy                 s_xy
        2.1755532033187186      2.5528312162500004      -3095.71337958186       -14334.175019905553     -266.6246571621359
        3.8539612787612816      2.5528312162500004      1984.545940564741       2600.022713916451       -231.51173473605857
        1.9487053877612812      2.6971687837500005      -2440.5325347603434     -12174.900189860842     218.16668098815774
        3.7297801301587183      2.6971687837500005      2346.884896581666       3783.157914612522       251.2555857653726
Element   4
--------------------
        x-coord                 y-coord                 s_xx                 s_yy                 s_xy
        1.6589745952456316      2.80283121625   -1450.2889582965072     -8872.074659345106      -72.06681066943815
```

```
        3.6057368370043683      2.80283121625    978.5767718222421
            -775.8555589492763       -113.15373316692695
        1.0942631625043682      2.94716878375    -860.4741921451364
            -6883.64755330906        95.91041299106614
        3.391025405245632       2.94716878375    1198.2604934500553
            -21.198601325088468      61.084662262566795
Element   5
----------------------------
        x-coord                 y-coord                  s_xx
                              s_yy                   s_xy
        0.8607695138596391      3.2113248650000004       -396.64924704039134
                -5356.253730106739       -143.5474018572677
        3.212435567140361       3.2113248650000004       442.6440969565484
                  -2558.6092501169396      -110.09640983575758
        0.7875644331403608      3.788675135      -342.44921755712755
            -5253.377292728361       120.33000626266096
        2.93923048585964        3.788675135      574.8574729154343
            -2195.6883244864875      156.89030905041264
Element   6
----------------------------
        x-coord                 y-coord                  s_xx
                              s_yy                   s_xy
        0.733974594859639       4.211324865      -304.920640016166
            -5202.146620276642       -89.02922430636458
        2.7392304861403614      4.211324865      52.88392532533228
            -4009.464735804978       -41.76416295720229
        0.6607695141403609      4.788675135      -247.22231378114938
            -5140.825389550244       46.39111510169215
        2.4660254048596393      4.788675135      150.22256717271668
            -3816.0091197040238      98.89256002670345
```

变形前和变形后的节点如图 2 所示，最大正应力的所在的位置和数值所在的范围如图 3 所示。

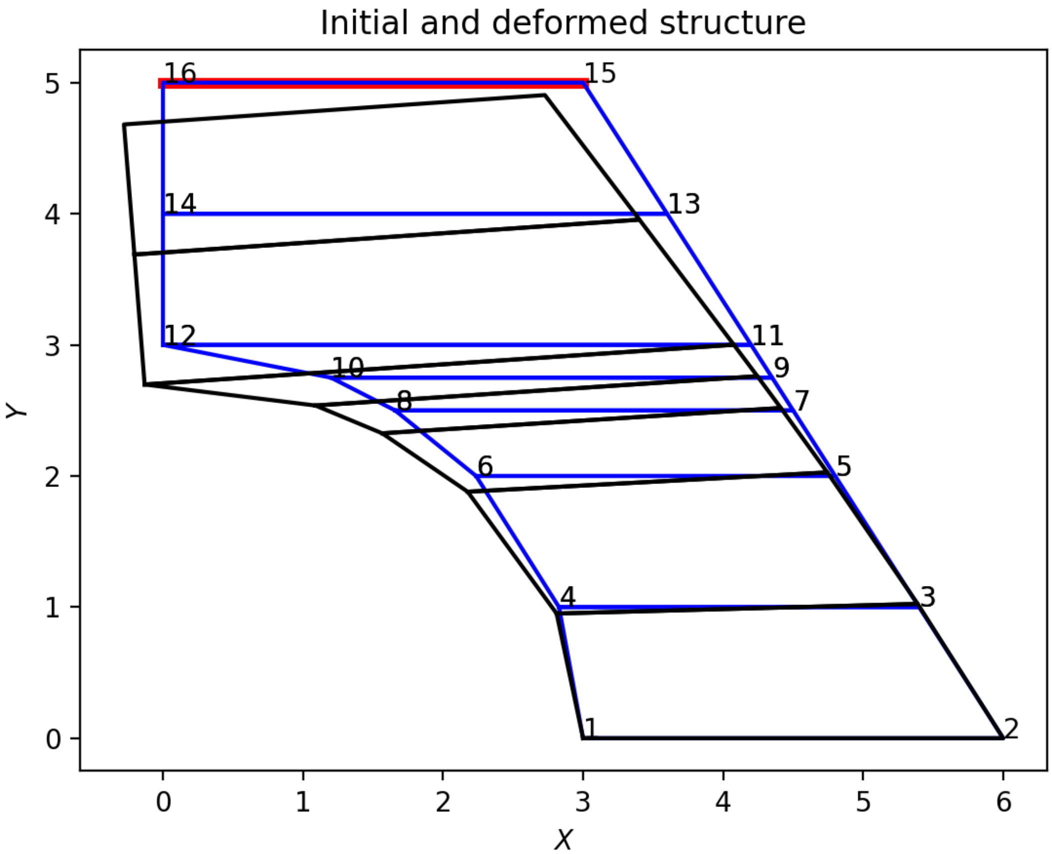


图 2: Initial and deformed structure

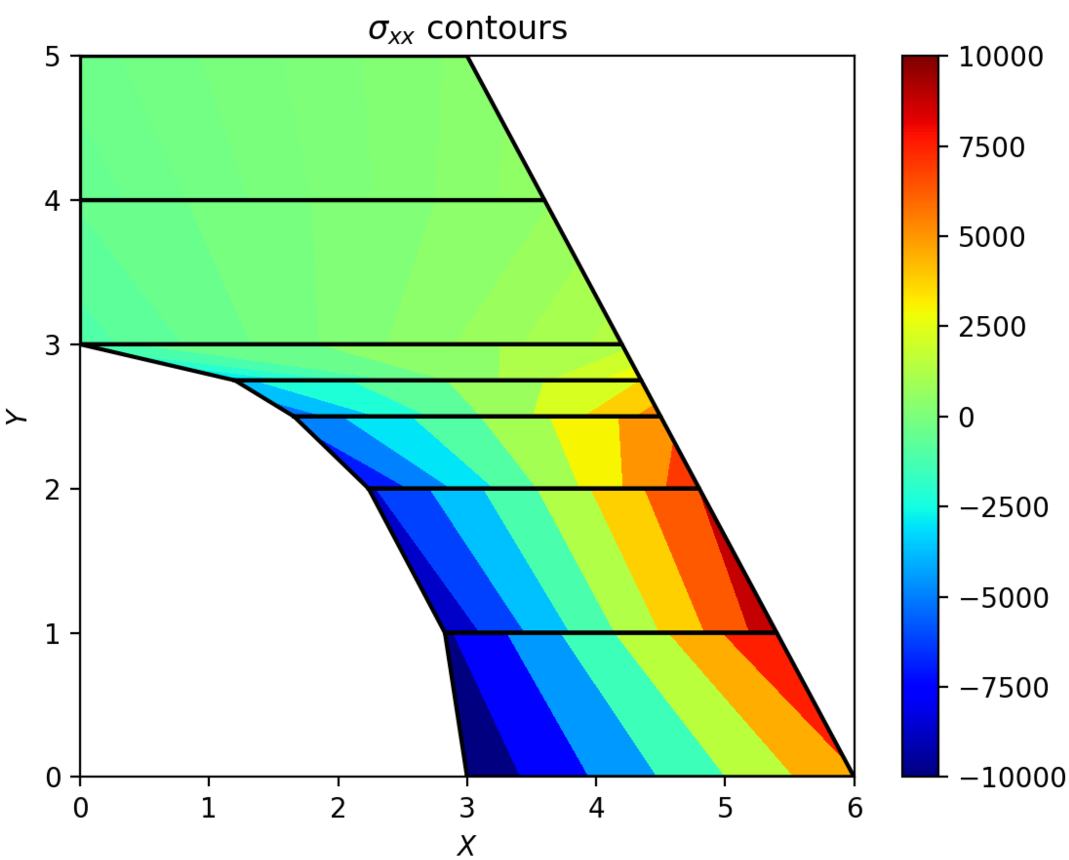


图 3: Stress Contours

2 习题 4.12

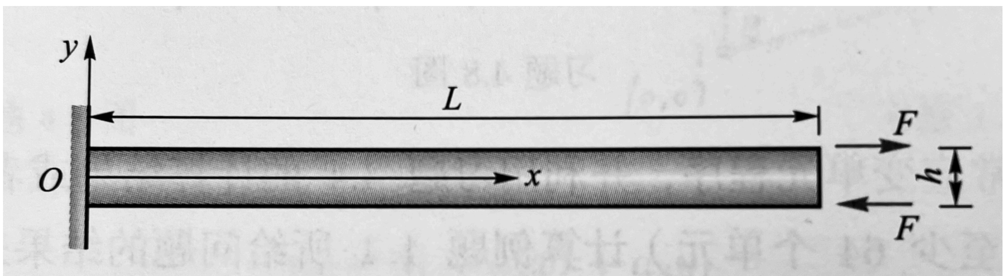
习题 4.12 图所示为一长 L=5.0，高 h=0.5，宽 b=0.1 的矩形截面悬臂梁，其左端固定，右端受一力偶（F=1）作用。取泊松比 $\nu = 0$，以模拟一维应力状态。位移和应力的解析解为

$$u(x) = \frac{Fh}{EI}xy, \quad v(x) = -\frac{Fh}{2EI}x^2, \quad \sigma_x(x) = \frac{Fh}{I}y$$

其中 $I = \frac{bh^3}{12}$ 为截面惯性矩。此问题可以用一平面应力问题模拟，弹性模量取为 E=10000。

(1) 将求解域分别用 1×5 和 2×10 个均匀规则单元进行离散，利用 elasticity2d-python 程序采用完全积分和缩减积分求解，画出用五种不同方案（解析解、网格一完全积分、网格一缩减积分、网格二完全积分、网格二缩减积分）计算得到的轴线（y=0）的挠度 v(x) 曲线和 $y = \frac{\sqrt{3}}{12}$(对应于高斯点 $\eta = \frac{\sqrt{3}}{3}$) 线上的正应力 $\sigma_x(x)$ 的分布，并分析。

(2) 使用 $N \times 5N$ 网格（N=1,2,4,8）研究有限元位移解的 L_2 范数和能量范数收敛率。



习题 4.12 图

2.1 解析求解

位移和应力的解析解为

$$u(x) = \frac{Fh}{EI}xy, \quad v(x) = -\frac{Fh}{2EI}x^2, \quad \sigma_x(y) = \frac{Fh}{I}y$$

其中 $I = \frac{bh^3}{12} = 1.04167 \times 10^{-3}$ 为截面惯性矩，L=5.0，高 h=0.5，宽 b=0.1。此问题可以用一平面应力问题模拟，弹性模量取为 E=10000。带入数值得到解析解为

$$u(x) = 0.048xy, \quad v(x) = -0.024x^2, \quad \sigma_x(y) = 480y$$

使用 python 程序绘制中线挠度和 $y = \frac{\sqrt{3}}{12}$(对应于高斯点 $\eta = \frac{\sqrt{3}}{3}$) 线上的正应力 $\sigma_x(x)$ 的分布如下：

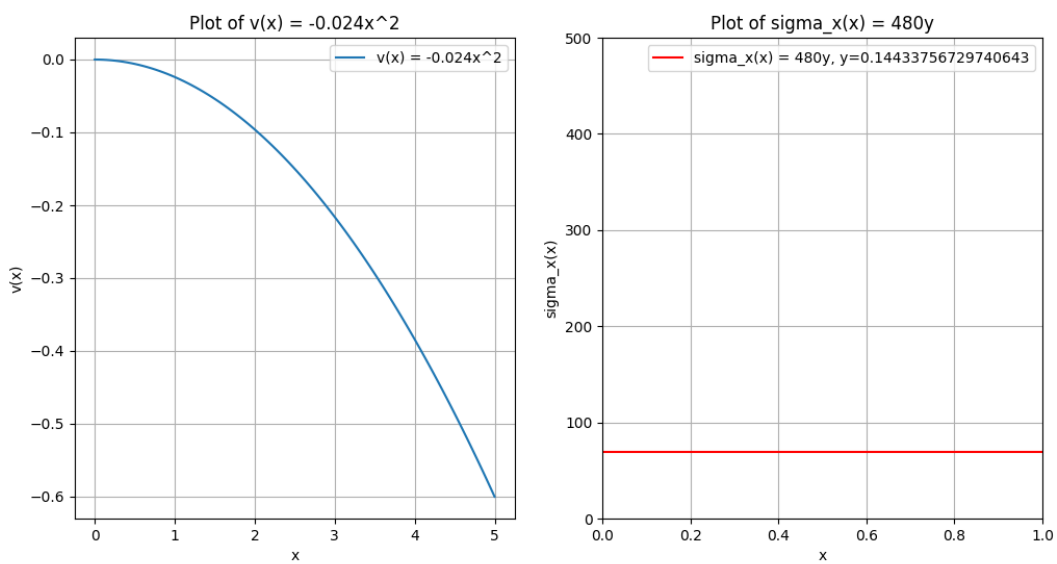


图 4: 解析挠度和正应力

2.2 1×5 单元完全积分求解

将求解域均匀划分为五个四节点四边形单元并求解，得到如下的结果

```
  Mesh Params
No. of Elements  5
No. of Nodes     12
No. of Equations 24

Condition number of stiffness matrix:  17957.835280242125

solution d
[ 0.      0.      0.      0.     -0.004  -0.008   0.004  -0.008  -0.008  -0.032
  0.008  -0.032  -0.012  -0.072   0.012  -0.072  -0.016  -0.128   0.016  -0.128
 -0.02   -0.2     0.02   -0.2   ]

reaction f =
 [[ 1.00000000e+01]
 [-3.01980663e-13]
 [-1.00000000e+01]
 [-3.05533376e-13]]
```

变形前后的模型如下图所示

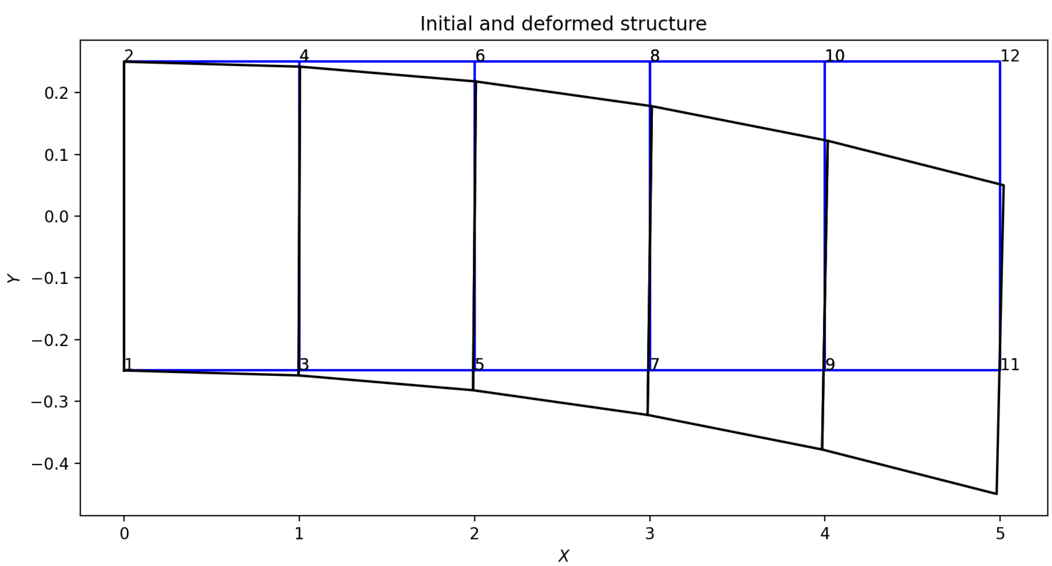


图 5: 5 单元网格变形前后（完全积分）

变形后的应力状态如图

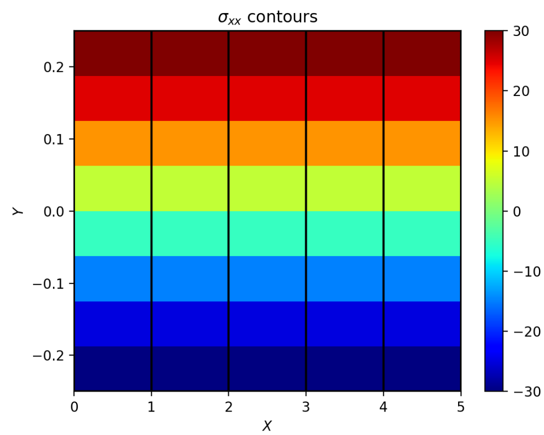


图 6: 1×5 网格应力状态（完全积分）

梁中性轴上的挠度和 $y = \frac{\sqrt{3}}{12}$(对应于高斯点 $\eta = \frac{\sqrt{3}}{3}$) 线上的正应力 $\sigma_x(x)$ 的分布如下，

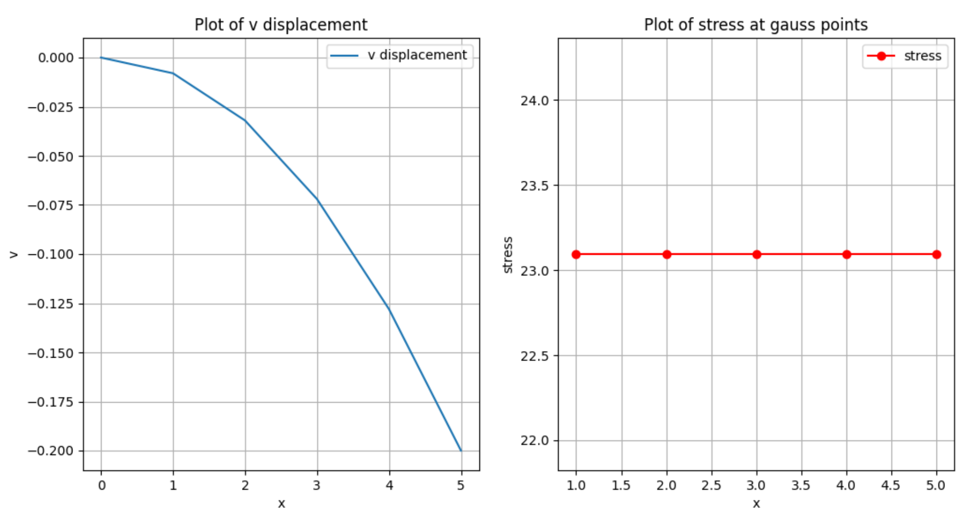


图 7: 1×5 网格挠度和正应力（完全积分）

2.3 1×5 单元缩减积分求解

修改沙漏控制的系数为 0.1，将求解域均匀划分为五个四节点四边形单元并求解，得到如下的结果

```
  Mesh Params
No. of Elements   5
No. of Nodes      12
No. of Equations 24

Condition number of stiffness matrix:  17783.97266935397

solution d
[ 0.      0.      0.      0.     -0.004  -0.008   0.004  -0.008  -0.008  -0.032
  0.008  -0.032  -0.012  -0.072   0.012  -0.072  -0.016  -0.128   0.016  -0.128
 -0.02   -0.2     0.02   -0.2   ]

reaction f =
 [[ 1.00000000e+01]
 [-1.42108547e-13]
 [-1.00000000e+01]
 [-1.38555833e-13]]
```

变形前后的模型如下图所示

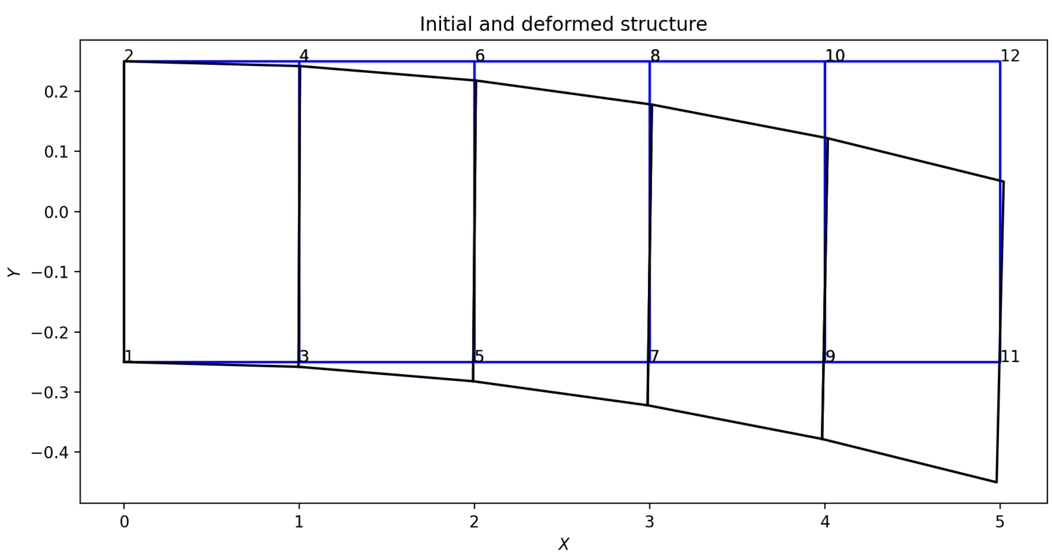


图 8: 5 单元网格变形前后（减缩积分）

变形后的应力状态如图

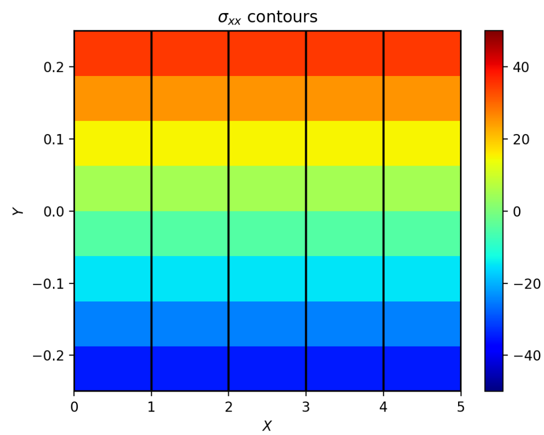


图 9: 1×5 网格应力状态（减缩积分）

梁中性轴上的挠度和 $y = \frac{\sqrt{3}}{12}$(对应于高斯点 $\eta = \frac{\sqrt{3}}{3}$) 线上的正应力 $\sigma_x(x)$ 的分布如下，

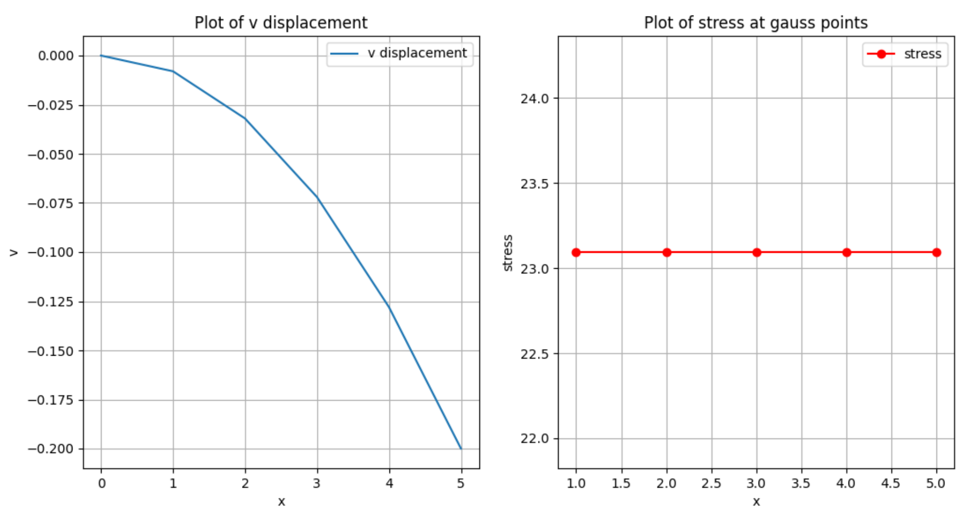


图 10: 1×5 网格挠度和正应力（减缩积分）

2.4 2×10 单元完全积分求解

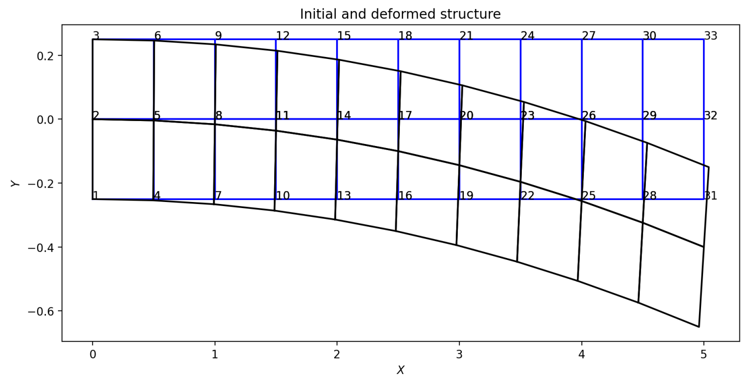
将求解域均匀划分为二个四节点四边形单元并求解，得到如下的结果

```
  Mesh Params
No. of Elements  20
No. of Nodes     33
No. of Equations 66

Condition number of stiffness matrix:  140975.53516224574

solution d
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00 -3.33333333e-04 -3.33333333e-04
 -2.37994273e-18 -3.33333333e-04  3.33333333e-04 -3.33333333e-04
 -6.66666666e-04 -1.33333333e-03 -4.52375547e-18 -1.33333333e-03
  6.66666666e-04 -1.33333333e-03 -9.99999999e-04 -3.00000000e-03
 -6.78585297e-18 -3.00000000e-03  9.99999999e-04 -3.00000000e-03
 -1.33333333e-03 -5.33333333e-03 -9.26984200e-18 -5.33333333e-03
  1.33333333e-03 -5.33333333e-03 -1.66666667e-03 -8.33333333e-03
 -1.08905004e-17 -8.33333333e-03  1.66666667e-03 -8.33333333e-03
 -2.00000000e-03 -1.20000000e-02 -1.32812022e-17 -1.20000000e-02
  2.00000000e-03 -1.20000000e-02 -2.33333333e-03 -1.63333333e-02
 -1.47147394e-17 -1.63333333e-02  2.33333333e-03 -1.63333333e-02
 -2.66666666e-03 -2.13333333e-02 -1.35903583e-17 -2.13333333e-02
  2.66666666e-03 -2.13333333e-02 -3.00000000e-03 -2.70000000e-02
 -1.23449282e-17 -2.70000000e-02  3.00000000e-03 -2.70000000e-02
 -3.33333333e-03 -3.33333333e-02 -1.12882668e-17 -3.33333333e-02
  3.33333333e-03 -3.33333333e-02]

reaction f =
 [[ 8.33333334e-01]
 [ 7.61057883e-14]
 [ 1.13797860e-14]
 [ 1.52100554e-13]
 [-8.33333334e-01]
 [ 7.57727214e-14]]
```



变形后的应力状态如图

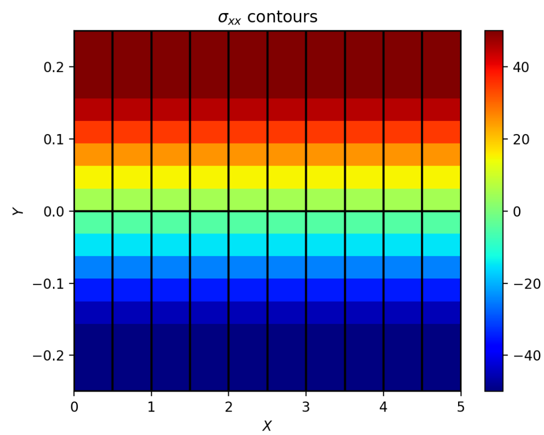


图 11: 2×10 网格应力状态（完全积分）

梁中性轴上挠度和 $y = \frac{\sqrt{3}}{12}$(对应于高斯点 $\eta = 0.1547$) 线上正应力 $\sigma_x(x)$ 的分布如下，

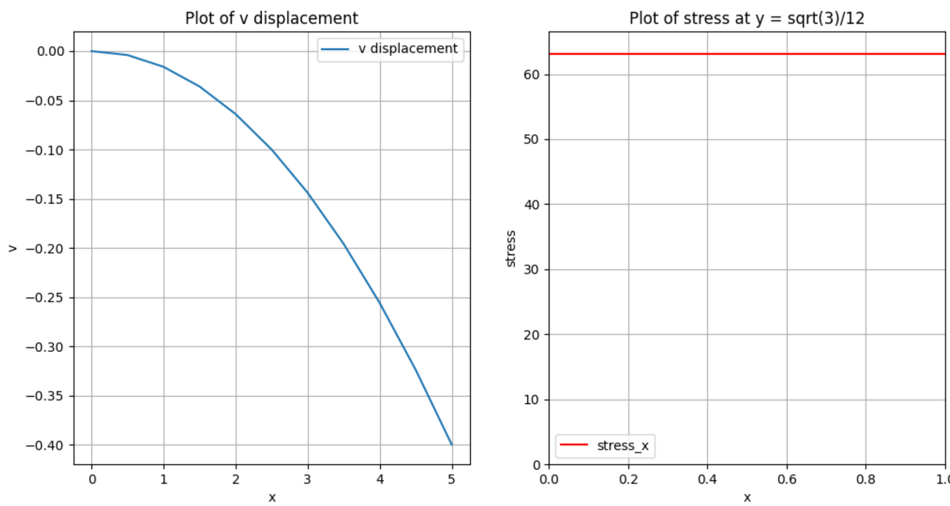


图 12: 2×10 网格挠度和正应力（完全积分）

这里为了求解 $y = \frac{\sqrt{3}}{12}$ 线上的正应力，我们计算了 $x > 0$ 部分的单元，$y = \frac{\sqrt{3}}{12}$ 在母空间中对应的 $\eta = 0.1547$，再使用如下程序进行计算和绘图：

```

def get_stress_at_line(e):
    """
    Print the element stress on y = sqrt(3)/12.

    Args:
        e    : The element number
    """
    de = model.d[model.LM[:, e] - 1]     # extract element nodal displacements

    # get coordinates of element nodes
    je = model.IEN[:, e] - 1
    C = np.array([model.x[je], model.y[je]]).T

    # compute strains and stresses at the element
    strain = np.zeros((3, 1))
    stress = np.zeros(3)
    eta = 0.1547 # for 2 * 10 mesh.
    psi = 1 # stress at certain y is the same through out the entire element, so x value does not matter

```

```python

    B, detJ = BmatElast2D(eta, psi, C)

    strain[:, 0] = (B @ de).T.squeeze()
    stress = (model.D @ (strain[:, 0].reshape((-1, 1)))).T.squeeze()

    print("\nStress_xx at y = sqrt(3)/12 :")
    print(stress)
    return stress

def get_disp_at_0():
    # get displacement from model.d
    v=np.zeros(11)
    for index in range(11):
        v[index] = model.d[(index+1) * 6 - 1]
    return v

def plot_stress_disp():
    # calculate the stress in an element(all stress are the same at same y coordinate)
    stress = get_stress_at_line(1)

    # calculate the disp at y=0
    v = get_disp_at_0()

    plt.figure(figsize=(12, 6))

    # plot disp
    x_disp=[0,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5]
    plt.subplot(1, 2, 1)
    plt.plot(x_disp, v, label='v displacement')
    plt.title('Plot of v displacement')
    plt.xlabel('x')
    plt.ylabel('v')
    plt.grid(True)
    plt.legend()

    # plot stress
    x_stress = np.linspace(0, 5, 100)
    plt.subplot(1, 2, 2)
    plt.axhline(y=stress[0], color='r', label=f'stress_x')
```

```
plt.title('Plot of stress at y = sqrt(3)/12')
plt.xlabel('x')
plt.ylabel('stress')
plt.ylim(0, 5)
plt.grid(True)
plt.legend()

plt.show()
```

2.5 2×10 单元减缩积分求解

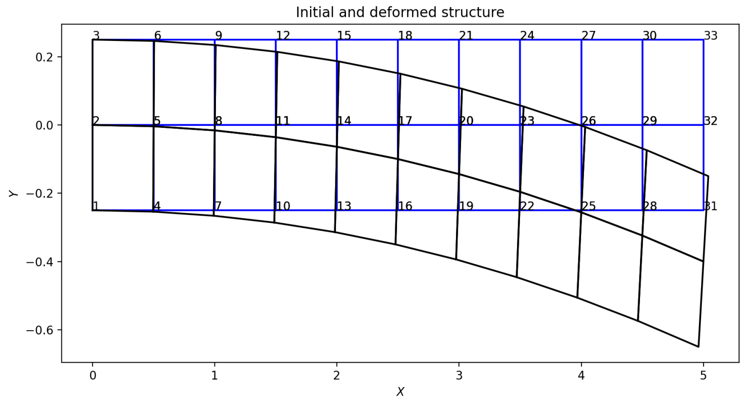
将求解域均匀划分为二个四节点四边形单元并求解，得到如下的结果

```
      Mesh Params
No. of Elements  20
No. of Nodes     33
No. of Equations 66

Condition number of stiffness matrix:  253219.82878534106

solution d
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00 -4.54545455e-04 -4.54545455e-04
 -2.65413989e-18 -4.54545455e-04  4.54545455e-04 -4.54545455e-04
 -9.09090909e-04 -1.81818182e-03 -5.21117945e-18 -1.81818182e-03
  9.09090909e-04 -1.81818182e-03 -1.36363636e-03 -4.09090909e-03
 -7.51368494e-18 -4.09090909e-03  1.36363636e-03 -4.09090909e-03
 -1.81818182e-03 -7.27272727e-03 -9.59269815e-18 -7.27272727e-03
  1.81818182e-03 -7.27272727e-03 -2.27272727e-03 -1.13636364e-02
 -1.07964426e-17 -1.13636364e-02  2.27272727e-03 -1.13636364e-02
 -2.72727273e-03 -1.63636364e-02 -1.14183509e-17 -1.63636364e-02
  2.72727273e-03 -1.63636364e-02 -3.18181818e-03 -2.22727273e-02
 -1.30486803e-17 -2.22727273e-02  3.18181818e-03 -2.22727273e-02
 -3.63636364e-03 -2.90909091e-02 -1.56878167e-17 -2.90909091e-02
  3.63636364e-03 -2.90909091e-02 -4.09090909e-03 -3.68181818e-02
 -1.23685814e-17 -3.68181818e-02  4.09090909e-03 -3.68181818e-02
 -4.54545455e-03 -4.54545455e-02 -2.07910476e-17 -4.54545455e-02
  4.54545455e-03 -4.54545455e-02]

reaction f =
 [[ 6.25000000e-01]
 [ 4.42978987e-14]
 [ 1.24344979e-14]
 [ 1.33670852e-13]
 [-6.25000000e-01]
 [ 4.59632332e-14]]
```



变形后的应力状态如图

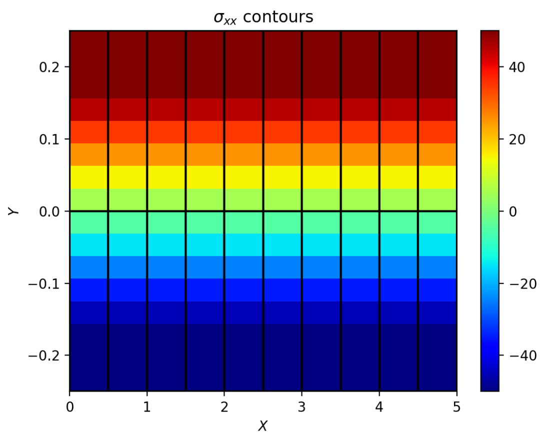


图 13: 2 × 10 网格应力状态（减缩积分）

梁中性轴上挠度和 $y=\frac{\sqrt{3}}{12}$(对应于高斯点 $\eta=0.1547$) 线上正应力 $\sigma_x(x)$ 的分布如下，

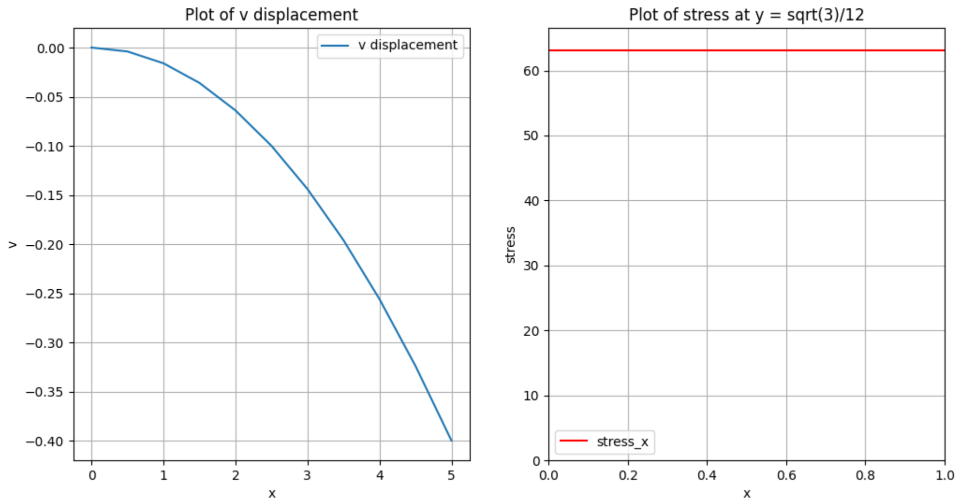


图 14: 2×10 网格挠度和正应力（减缩积分）

2.6 分析

查看以上结果发现，有限元计算的结果和精确解相比相差较大，但是随着网格的加密，误差有减小的趋势。并且使用减缩积分和完全积分相比，二者的结果几乎完全相同，这也是由于我们所采用的网格是规则、边界为直线的网格。

2.7 有限元位移解的 L_2 范数和能量范数

要研究有限元位移解的范数收敛率，我们需要绘制范数-单元特征长度的双对数曲线。首先完成四种网格有限元解的计算，再计算对应的范数，最后绘制曲线。该习题中我们将问题简化为了平面应力问题，单元特征长度我们取为单元的对角线长度。曲线如下：

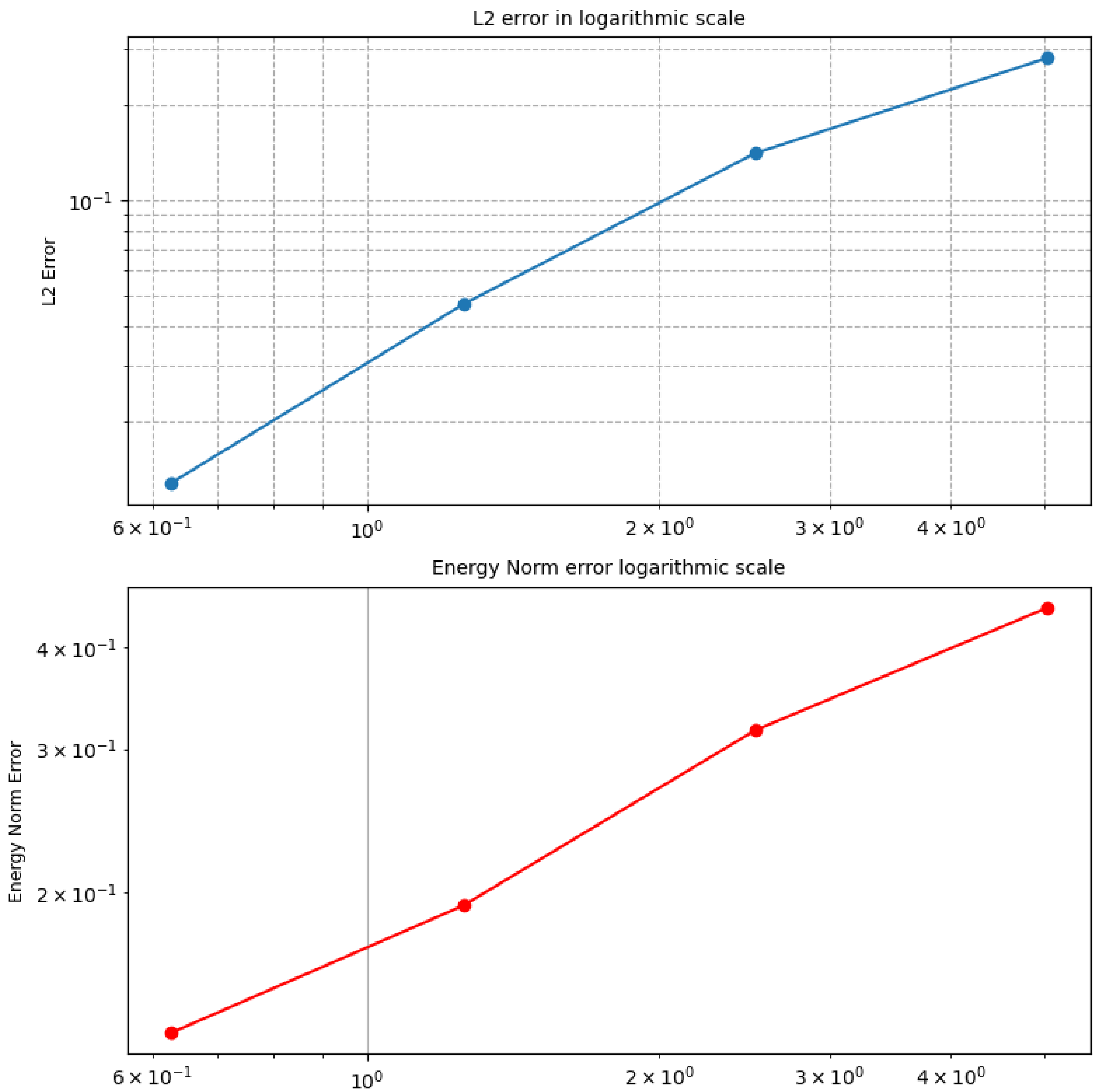


图 15: 误差-单元特征长度双对数曲线

程序拟合输出了两条曲线的斜率（即收敛率）：

```
L2 Error Linear Regression (log-log scale): Slope = 1.4971236161252606,
    Intercept = 0.029583819983313876
Energy Norm Error Linear Regression: Slope = 0.5920973705203383, Intercept =
    -1.7429697765590348
```

理论上，L2 范数的收敛率应为 2，能量范数的收敛率应为 1。但是本实验中由于开始使用的网格过于稀疏，产生了很大的误差，最终导致收敛率的斜率较理论值偏低。

计算范数误差的函数如下：

```python
  def ErrorNorm_4_12():
"""
Calculate and print the error norm (L2 norm) of exercise 4.12

Return the L2Norm Error and h
"""

ngp = 2
w, gp = gauss(ngp)    # extract Gauss points and weights

beam_height = 0.5       # geo pramras of the beam
beam_length = 5

L2Norm = 0

EnergyNorm = 0


# compute the L2 and Energy error norm element-wise
for e in range(model.nel):
  # get coordinates of element nodes
  je = model.IEN[:, e] - 1
  C = np.array([model.x[je], model.y[je]]).T
  # extract element nodal displacements
  de = model.d[model.LM[:, e] - 1]
  # compute the L2Norm = uex - uh gauss intergration
  # here we calculate the displacement field in parent space first then
                                    using Jacobian to transfer
  for i in range(ngp):
    for j in range(ngp):
      eta = gp[i]
      psi = gp[j]
```

```python
            Be, detJe = BmatElast2D(eta, psi, C)
            Ne = NmatElast2D(eta, psi)

            # transfer parent coordinate to physical for exact displacement
            x_exact = 0.25*(1-psi)*(1-eta) * C[0][0] + 0.25*(1+psi)*(1-eta) * C[1][0] + 0.25*(1+psi)*(1+eta) * C[2][0] + 0.25*(1-psi)*(1+eta) * C[3][0]
            y_exact = 0.25*(1-psi)*(1-eta) * C[0][1] + 0.25*(1+psi)*(1-eta) * C[1][1] + 0.25*(1+psi)*(1+eta) * C[2][1] + 0.25*(1-psi)*(1+eta) * C[3][1]

            # for 2d problem, displacement field is a 2*1 vector
            u, v, sxx= exact_4_12(x_exact, y_exact)
            uh = Ne @ de # displacement in parent space
            error_x = uh[0] - u
            error_y = uh[1] - v
            Nabla_u = np.array([sxx, 0.0, 0.0])

            L2Norm +=  w[i] * w[j] * (np.power(error_x, 2) + np.power(error_y, 2)) * detJe
            EnergyNorm += w[i] * w[j] * (Nabla_u - Be @ de).T @ model.D @ (Nabla_u - Be @ de) * detJe

    # L2Norm is the sqrt of the integral
    L2Norm = np.sqrt(L2Norm)
    # EnergyNorm is 1/2 and sqrt of the a
    EnergyNorm = 0.5 * np.sqrt(EnergyNorm)

    # diagonal_length h
    N = np.sqrt(model.nel/5)
    h = np.sqrt((beam_height/N)**2 + (beam_length/N)**2)

    # print Error norms
    print('\nError norms')
    print('%13s %13s %13s '
        %('h','L2Norm','EnergyNorm'))
    print('%13.6E %13.6E %13.6E\n'
        %(h, L2Norm, EnergyNorm))

    return h, L2Norm, EnergyNorm
```

主程序如下:

```python
    import numpy as np
import matplotlib.pyplot as plt
from Elasticity2D import FERun
from Exact import ErrorNorm_4_12

# Json data files for meshes
files = ("./Convergence/exercise_4_12_5Elem.json",
         "./Convergence/exercise_4_12_20Elem.json",
         "./Convergence/exercise_4_12_80Elem.json",
         "./Convergence/exercise_4_12_320Elem.json")



# Run FE analysis for all files using meshed
n = len(files)
h = np.zeros(n)
L2Norm = np.zeros(n)
EnergyNorm = np.zeros(n)
for i in range(n):
    FERun(files[i])

    # Calculate error norms for convergence study
    h[i], L2Norm[i], EnergyNorm[i] = ErrorNorm_4_12()


fig, axs = plt.subplots(2, 1, figsize=(8, 8))

# L2-Norm error and h figure
axs[0].set_title('L2 error in logarithmic scale', fontsize=10)
axs[0].set_ylabel('L2 Error', fontsize=9)
axs[0].set_xscale('log')
axs[0].set_yscale('log')
axs[0].plot(h, L2Norm, marker='o')
axs[0].grid(True, which="both", ls="--")
slope_L2, intercept_L2 = np.polyfit(np.log(h), np.log(L2Norm), 1)
print(f"L2 Error Linear Regression (log-log scale): Slope = {slope_L2},
                                            Intercept = {np.exp(intercept_L2)}")

# Energy Norm error and h figure
axs[1].set_title('Energy Norm error logarithmic scale', fontsize=10)
axs[1].set_ylabel('Energy Norm Error', fontsize=9)
```

```
axs[1].set_xscale('log')
axs[1].set_yscale('log')
axs[1].plot(h, EnergyNorm, marker='o', color='red')
axs[1].grid(True)
slope_Energy, intercept_Energy = np.polyfit(np.log(h), np.log(EnergyNorm), 1)
print(f"Energy Norm Error Linear Regression: Slope = {slope_Energy}, Intercept = {intercept_Energy}")

plt.tight_layout()
plt.show()
```