CEE 471 – Structural Mechanics

Term Project 2

Eric Jenkins

Abstract

This study addresses the goal of using finite elements to approximate the behavior of a stretched membrane subject to lateral loads.

Introduction

Computational structural analysis has become an extremely important link in the safe design of buildings, bridges, and other engineering-related structures. Possibly the most popular method of computational analysis is the Finite Element Method (FEM). This study considers the use of FEM on various membrane structures subject to lateral loads.

Analysis Principles

FEM originated from the need to solve complex elasticity problems in structural mechanics and also aeronautical engineering. Its birth can be traced back to work done by A. Hrennikoff (1941) and R. Courant (1942), both of whom addressed the issue of analyzing a continuous domain by discretizing it into a set of discrete sub-domains.

FEM, as it relates to mesh analysis, draws from Partial Differential Equation (PDE) results laid down by the principles of the Ritz Method. The FEM subdivides a continuous structural mesh into smaller elements that can be represented linearly. When a mesh is defined, boundaries are set so that the simulated membrane will act as an elastically-deforming object and not a rigid body.

Consider a rectangular membrane with side lengths of L_1 and L_2 simply supported along the perimeter as shown here:

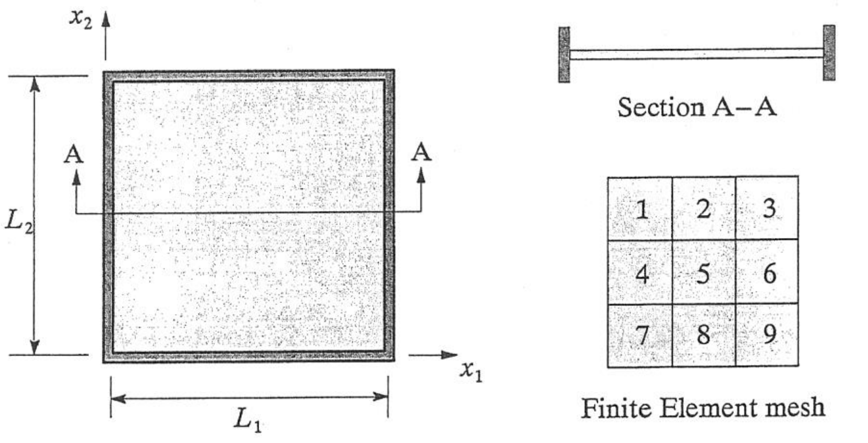


Figure 1. A sample membrane structure and its equivalent 3x3 finite element mesh

The governing differential equation is

$$div(T\,\nabla u) + p = 0$$

where u(x_1,x_2) is the transverse displacement field in the membrane, *p(x_1, x_2)* is the applied transverse load, and *T* is the (known) membrane tension. With shear and bending deformations negligible for the membrane, the governing differential equation is given by

$$G(u,\bar{u}) \equiv \int_{\Omega} [T\,\nabla u \cdot \nabla \bar{u} - p\bar{u}]dA = 0, \qquad \text{for all } \bar{u} \in \mathrm{P}_e(\Omega)$$

First, Ritz approximation will be performed, which will later be used for validation of the FEM approximation. Let the displacement and the virtual displacement fields $u(x), \bar{u}(x)$ be approximated by three-dimensional vector base functions $\{h_1(x),...,h_N(x)\}$ as

$$u(x) = \sum_{n=1}^{N} a_n h_n(x)\,, \quad u(x) = \sum_{n=1}^{N} \bar{a}_n h_n(x)$$

Then the stiffness matrix K and force vector f have components

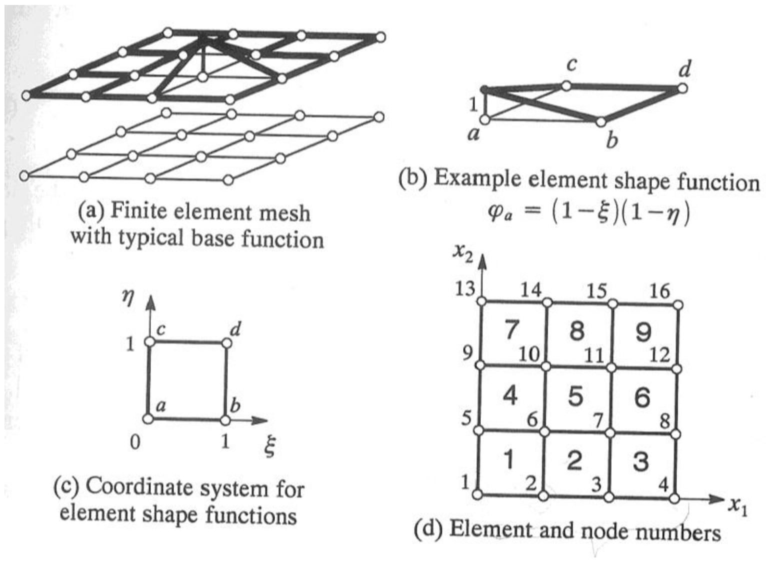
$$K_{ij} = \int_{\Omega} T\nabla h_i \cdot \nabla h_j dA\,, \qquad f_i = \int_{\Omega} ph_i\; dA$$

The Ritz approximation leads to the usual discrete version of the functional G given by

$$G(a,\bar{a}) = \bar{a}^T (Ka - f)$$

suggesting that a represents an equilibrium configuration if and only if $G(a,\bar{a}) = 0$ for all $\bar{a}$. This, in turn, implies that *Ka = f*, where *K* is the stiffness matrix and the vector *f* is the load vector.

Consider the following square membrane divided into 9 elements and 16 nodes:



(a) Finite element mesh with typical base function
(b) Example element shape function $\varphi_a = (1-\xi)(1-\eta)$
c
d
1
a
b
(c) Coordinate system for element shape functions
η
ξ
0
1
(d) Element and node numbers
x_2
x_1
13
14
15
16
9
10
11
12
5
6
7
8
1
2
3
4

Figure 2. Illustration of the finite element basis and membrane shape functions

The four element shape functions for two dimensional problems are

$$\varphi_1 = (1-\xi)(1-\eta) \qquad \varphi_2 = \xi(1-\eta)$$
$$\varphi_3 = (1-\xi)\eta \qquad \varphi_4 = \xi\eta$$

The element domain can be mapped to the unit square through a bilinear function, with the change of variable expressed as

$$\xi = \frac{x_1 - x_2^a}{x_1^b - x_1^a}, \qquad \eta = \frac{x_2 - x_2^a}{x_2^c - x_2^a}, \qquad dx_1 dx_2 = \ell_1 \ell_2 d\xi d\eta$$

where l_1 and l_2 are the actual element dimensions. Letting $\varphi \equiv [\varphi_1, \varphi_2, \varphi_3, \varphi_4]^T$, the real and virtual displacement fields can be written as

$$u_e = \varphi^T \mathrm{B}_e^T a, \qquad \bar{u}_e = \varphi^T \mathrm{B}_e^T \bar{a}$$

where $a \equiv [a_1, a_2, a_3, a_4]^T$ and $\bar{a} \equiv [\bar{a}_1, \bar{a}_2, \bar{a}_3, \bar{a}_4]^T$ are arrays containing the nodal unknowns and their virtual counterparts and

$a_e \quad b_e \quad c_e \quad d_e$

$$B_e^T \equiv \begin{bmatrix} 0 & \dots & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & 1 & \dots & 0 \end{bmatrix}$$

is a 4xN matrix whose purpose is to pick out the elements of the global vector that are associated with the four nodes of element *e*.

The gradient of the element function can be written as

$$\nabla u_e = \mathrm{B}^T(\varphi)B_e^T a, \qquad \mathrm{B}^T(\varphi) = J[\nabla\varphi_1, \nabla\varphi_2, \nabla\varphi_3, \nabla\varphi_4]$$

where $\nabla\varphi_i = [\partial\varphi_i / \partial\xi, \partial\varphi_i / \partial\eta]^T$ and the Jacobian of the change of variables is

$J = \left[diag[\ell_1 \ell_2]\right]^{-1}$. The virtual displacement gradient is of the same form.

The virtual work functional can now be written as

$$G(u, \bar{u}) \equiv \sum_{e=1}^{M} \int_0^1 \int_0^1 (T\nabla u_e \cdot \nabla\bar{u}_e - p\bar{u}_e) J d\xi d\eta$$

where $\hat{J} \equiv \ell_1 \ell_2$. Finally, we can now define the element stiffness matrix and force vector as

$$k_e \equiv \int_0^1 \int_0^1 T\mathrm{B}(\varphi)\mathrm{B}(\varphi)^T \hat{J} d\xi d\eta, \qquad f_e \equiv \int_0^1 \int_0^1 p\varphi\hat{J} d\xi d\eta$$

where the element stiffness matrix K_e is 4x4 and the force vector f_e is 4x1. From here, the local stiffness matrices and force vectors for each element are assembled into the global stiffness matrix, K, and the global force vector, f, respectively.

Nodal displacements that occur as a result of the transverse loading can be represented as:

$$\begin{bmatrix} f_f \\ f_s \end{bmatrix} = \begin{bmatrix} K_{ff} & K_{fs} \\ K_{sf} & K_{ss} \end{bmatrix} \begin{bmatrix} u_f \\ u_s \end{bmatrix}$$

where f_f and u_f represent the free node applied forces and displacements, respectively, and f_s and u_s represent the boundary node reaction forces and prescribed displacement, respectively. The K matrix is partitioned to satisfy the stiffness values associated with each node of each element.

Knowing the applied load at all the free nodes f_f and the boundary displacements u_s, the resulting free node displacements can be obtained by solving the identity equation $f_f = K_{ff}u_f + K_{fs}u_s$ for u_f.

Once u_f is known, the reaction forces f_s can be solved for by carrying out the identity equation $f_s = K_{sf}u_f + K_{ss}u_s$.

Computer Implementation

This study considers the discretization of a continuous membrane structure with tension T and applied transverse loading $p(x_1, x_2)$ into rectangular elements of side lengths l_1 and l_2. All membranes to be considered in this study are planar in (x_1,x_2) space, with transverse loads, resultant displacements, and reaction forces all in the x_3 plane.

The Matlab FEM code for this project has several key features that attempt to make the tasks of element definition, load definition, boundary node definition, and resultant plotting as automated as possible. This Matlab program has the ability to perform rectangular-element mesh analysis for a uniform-tension membrane of any shape (including holes), with virtually any element-wise loading conditions the user might desire.

Simple Analysis and Verification

The first analysis task of this study was to verify the program's functionality by analyzing a simple square membrane of unit length subjected to uniform load p=1. First, the membrane was discretized into a 3x3 mesh, as shown in Figure 2(d). The resulting deformation plot was as such:

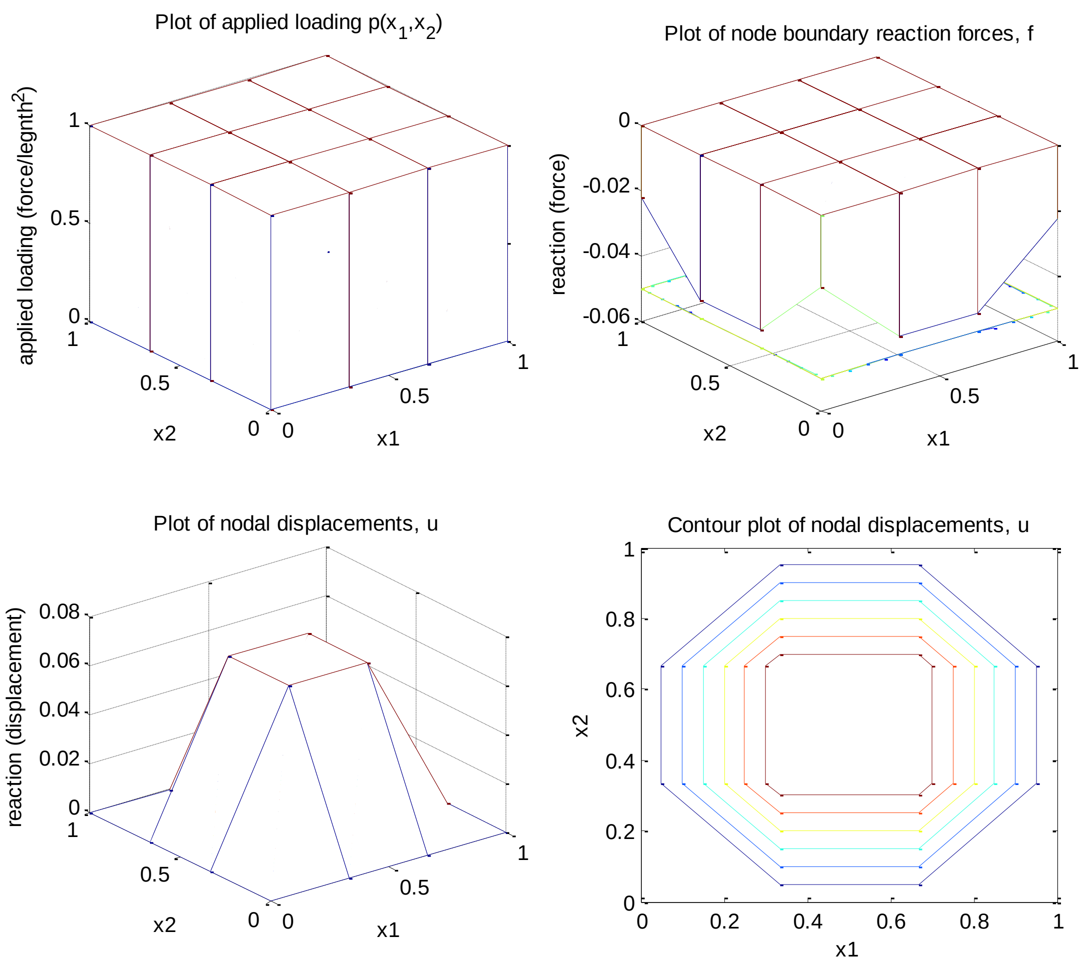


Figure 3. Results of a 3x3 mesh analysis - square membrane subject to a uniform load p = 1.

Next, the same problem was considered under different load distributions, as shown in Appendix B. As it turns out, the membrane could not sustain a point load. Refining the mesh around a singular element point loading caused the resulting deformation to shrink by increasing orders of magnitude, and convergence was never reached.

The program was then validated against the Ritz approximation for a square membrane with the same properties and loading, as shown in Appendix A. The FEM approximation for maximum displacement was extremely close to even the second-order iteration Ritz approximation. However, the performance difference between the FEM program and the Ritz program was striking. The Ritz program slowed beyond usability at the fifth iteration. Of course, this is why high-order Ritz analysis is avoided in computational analysis.

Finally, the Matlab code was stretched to it limits by testing varying membrane shapes and loading conditions. Membrane shapes of all shapes and sizes were simulated, including triangles, circles, and shapes with holes. Discontinuous loading conditions such as point loads and stair-step loads were considered, as well as uniformly distributed loads. Appendix B contains the many varying cases that were considered.

Much of this was possible due to discontinuous plotting algorithms as well as algorithms to automatically traverse the membrane and number elements, no matter what the shape. Boundary nodes are automatically detected and numbered accordingly, even in the case of custom-drawn membrane shapes, such as the detailed example in Appendix C.

Conclusion

The FEM proves to be an extremely powerful tool for analyzing various mesh structures subject to lateral loading. Being that FEM uses only a one-term approximation for each of the discretized elements in a continuous structure, the computational expense of using FEM to analyze mesh structures is magnitudes less than that of using high-order Ritz approximations on the same continuous structures. Of course, three-dimensional finite element interpolation is still very computationally expensive for large-scale structures with great numbers of elements. My computer struggled with meshes of more than 5000 elements. This number sounds large, but when dealing with the simulated response of complex bridges or buildings, refinement of orders of magnitude higher is required.

I would like to see further study conducted on the computational cost vs. accuracy of using a two-term approximation in finite elements. A two-term Ritz approximation is much more accurate than the one-term approximation, but the two-term takes roughly four times as long to compute. However, if two-term approximation is used in place of one-term approximation, less elements would be required for an accurate reproduction of the mesh, and elements would then be treated as sinusoidal elements rather than linear ones.

The concept of using two-term approximations in place of one-term approximation is the 3-dimensional equivalent to using quadratic interpolation in place of linear interpolation when plotting in two dimensions. This study has been extremely insightful, and deeper analysis will hopefully prove equally fulfilling.

References

Hjelmstad, K.D. "Fundamentals of Structural Mechanics, 2nd Ed." Springer, New York. 2005.

Wikipedia. "Finite Element Method." http://en.wikipedia.org/wiki/Finite_element_method

Appendix A: Validation of FEM Matlab code using Ritz

The following validation script, validation.m, performs multi-term Ritz analysis on a square membrane of unit length, simply supported on all four boundaries with tension T=1, subjected to a uniform load of p=1.

The following is the output code for validation.m using N = 1 term:

```
% Output for N = 1 :
% For N = 1 terms: u(0.5,0.5) = 0.0821279
```

The following is the output code for validation.m using N = 3 terms:

```
% Output for N = 3 :
% For N = 3 terms: u(0.5,0.5) = 0.0721914
```

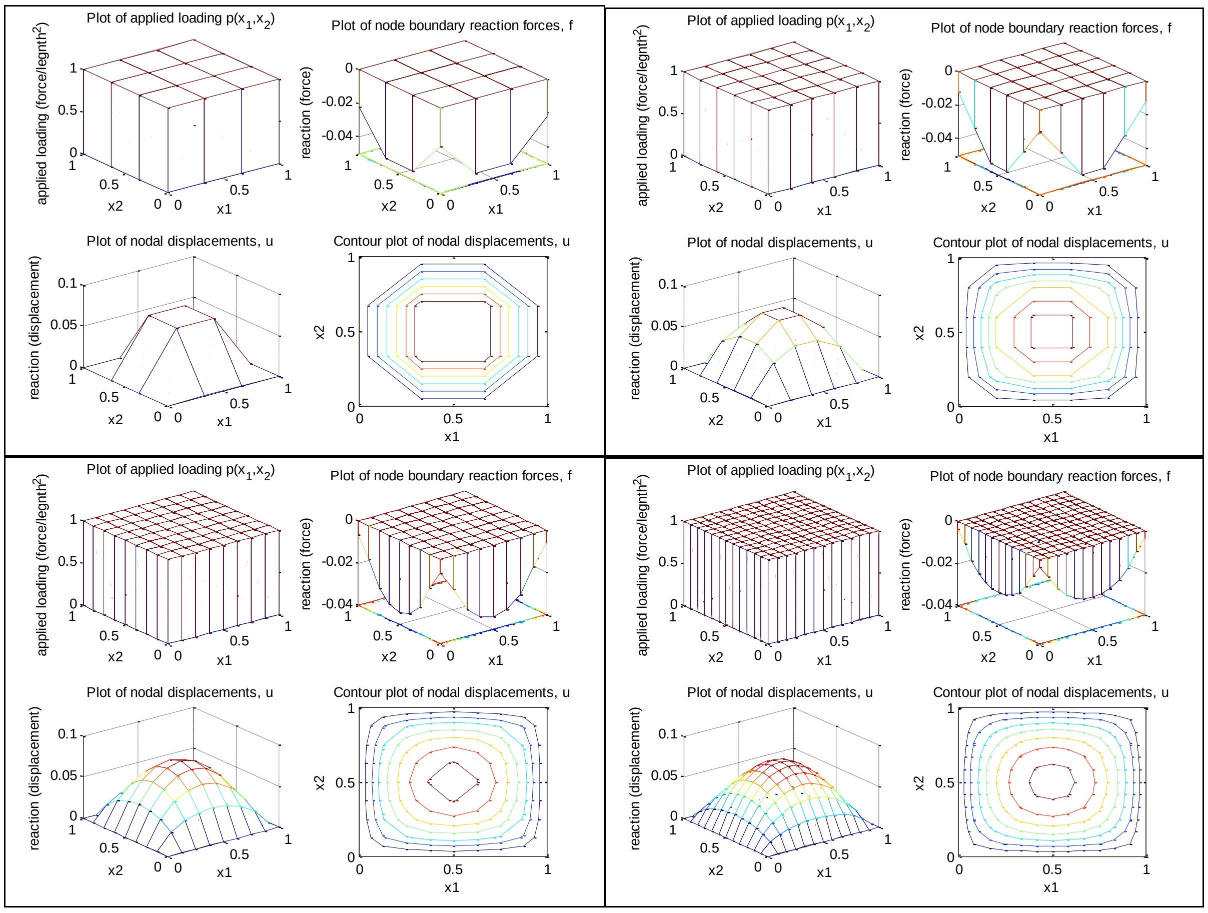
The validation program slows dramatically for analyses using any more than 3 terms. However, having patiently run cases for N = 4 and N = 5 terms, the maximum displacement for a unit length square membrane subjected to uniform unit loading converges to 0.073.

Next, an iterative finite element analysis is performed using the Matlab code, defined for the same unit-length square membrane, with p=1 and T=1.

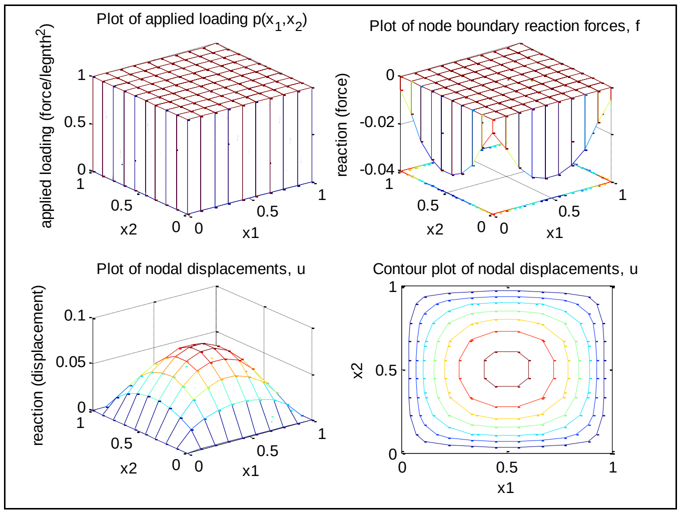
```
-------------------------------

Iteration 1:

Analysis on a 3 x 3 grid (9 elements):

Maximum displacement = 0.0666667

-------------------------------

Iteration 2:

Analysis on a 5 x 5 grid (25 elements):

Maximum displacement = 0.0710526

Percent change in max displacement =  -6.6 %

-------------------------------

Iteration 3:

Analysis on a 8 x 8 grid (64 elements):
```

```
Maximum displacement = 0.0745983
Percent change in max displacement =    -5 %
-------------------------------
Iteration 4:
Analysis on a 12 x 12 grid (144 elements):
Maximum displacement = 0.0740783
Percent change in max displacement =   0.7 %
 ... Done.
```

Mesh analysis at first four iterations:

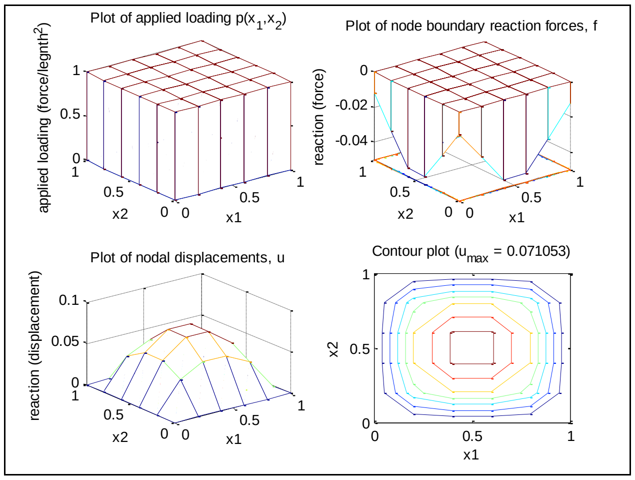


Mesh analysis at final iteration:

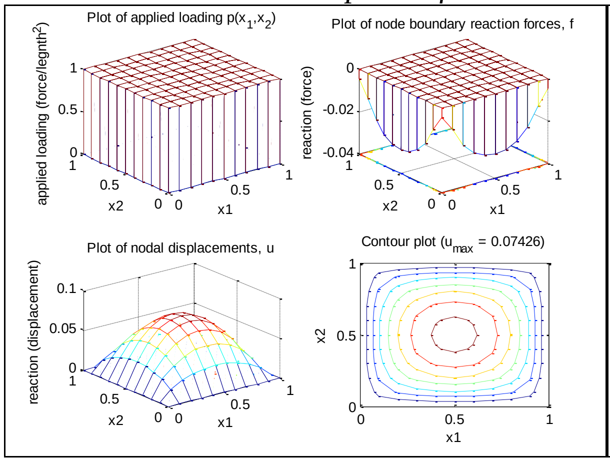


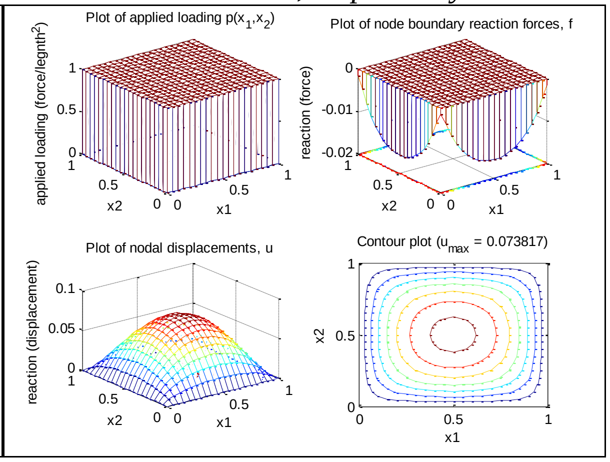
After proper convergence, the square shape was re-analyzed for a 5x5 mesh:

```
Analysis on a 5 x 5 grid (25 elements):
meshsize =
     5
elemL1 =
    0.2000
elemL2 =
    0.2000

Finite element mesh, with node & element labels (nodalmatrix, elementmatrix):

        x2
        ^
        |
 1.00  31------32------33------34------35------36
        |       |       |       |       |       |
        |  21   |  22   |  23   |  24   |  25   |
        |       |       |       |       |       |
 0.80  29------13------14------15------16------30
        |       |       |       |       |       |
        |  16   |  17   |  18   |  19   |  20   |
        |       |       |       |       |       |
 0.60  27------ 9------10------11------12------28
        |       |       |       |       |       |
        |  11   |  12   |  13   |  14   |  15   |
        |       |       |       |       |       |
 0.40  25------ 5------ 6------ 7------ 8------26
        |       |       |       |       |       |
        |   6   |   7   |   8   |   9   |  10   |
        |       |       |       |       |       |
 0.20  23------ 1------ 2------ 3------ 4------24
        |       |       |       |       |       |
        |   1   |   2   |   3   |   4   |   5   |
        |       |       |       |       |       |
 0.00  17------18------19------20------21------22 --> x1

      0.00    0.20    0.40    0.60    0.80    1.00

Node coordinates (nodalcoords):

| node| (   x1,    x2) |
|   1 | ( 0.20, 0.20) |
|   2 | ( 0.40, 0.20) |
|   3 | ( 0.60, 0.20) |
|   4 | ( 0.80, 0.20) |
|   5 | ( 0.20, 0.40) |
|   6 | ( 0.40, 0.40) |
|   7 | ( 0.60, 0.40) |
|   8 | ( 0.80, 0.40) |
|   9 | ( 0.20, 0.60) |
|  10 | ( 0.40, 0.60) |
|  11 | ( 0.60, 0.60) |
|  12 | ( 0.80, 0.60) |
|  13 | ( 0.20, 0.80) |
|  14 | ( 0.40, 0.80) |
|  15 | ( 0.60, 0.80) |
|  16 | ( 0.80, 0.80) |
|  17 | ( 0.00, 0.00) |
|  18 | ( 0.20, 0.00) |
|  19 | ( 0.40, 0.00) |
|  20 | ( 0.60, 0.00) |
|  21 | ( 0.80, 0.00) |
|  22 | ( 1.00, 0.00) |
|  23 | ( 0.00, 0.20) |
|  24 | ( 1.00, 0.20) |
|  25 | ( 0.00, 0.40) |
|  26 | ( 1.00, 0.40) |
|  27 | ( 0.00, 0.60) |
|  28 | ( 1.00, 0.60) |
|  29 | ( 0.00, 0.80) |
|  30 | ( 1.00, 0.80) |
|  31 | ( 0.00, 1.00) |
|  32 | ( 0.20, 1.00) |
|  33 | ( 0.40, 1.00) |
|  34 | ( 0.60, 1.00) |
|  35 | ( 0.80, 1.00) |
|  36 | ( 1.00, 1.00) |

Element connectivity matrix (elemnodes):
|       |      local node           |
|element|   a |   b |   c |   d |
|      1 |  17 |  18 |  23 |   1 |
|      2 |  18 |  19 |   1 |   2 |
|      3 |  19 |  20 |   2 |   3 |
|      4 |  20 |  21 |   3 |   4 |
|      5 |  21 |  22 |   4 |  24 |
|      6 |  23 |   1 |  25 |   5 |
|      7 |   1 |   2 |   5 |   6 |
|      8 |   2 |   3 |   6 |   7 |
|      9 |   3 |   4 |   7 |   8 |
|     10 |   4 |  24 |   8 |  26 |
|     11 |  25 |   5 |  27 |   9 |
|     12 |   5 |   6 |   9 |  10 |
|     13 |   6 |   7 |  10 |  11 |
|     14 |   7 |   8 |  11 |  12 |
|     15 |   8 |  26 |  12 |  28 |
|     16 |  27 |   9 |  29 |  13 |
|     17 |   9 |  10 |  13 |  14 |
|     18 |  10 |  11 |  14 |  15 |
|     19 |  11 |  12 |  15 |  16 |
|     20 |  12 |  28 |  16 |  30 |
|     21 |  29 |  13 |  31 |  32 |
|     22 |  13 |  14 |  32 |  33 |
|     23 |  14 |  15 |  33 |  34 |
|     24 |  15 |  16 |  34 |  35 |
|     25 |  16 |  30 |  35 |  36 |


Maximum displacement = 0.0710526
ans =
    0.0711
```



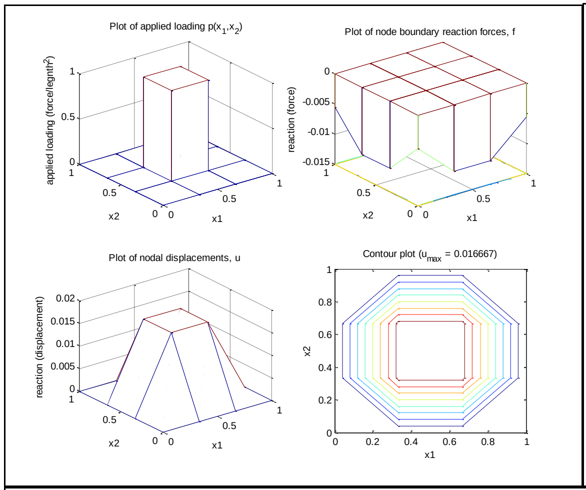
The same case was then repeated for a 10x10 mesh and a 20x20 mesh, respectively:

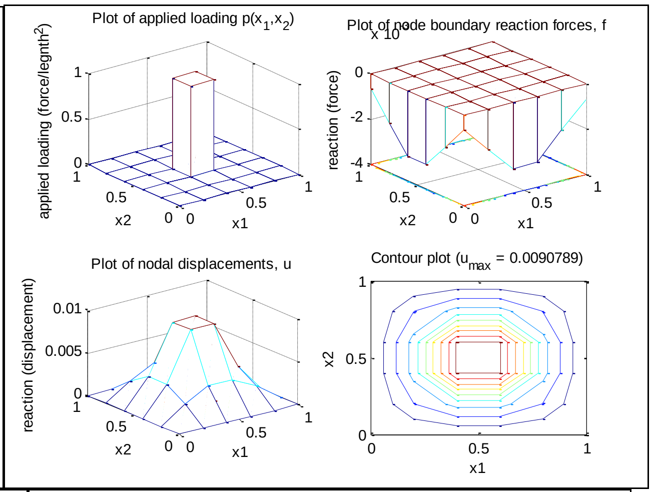


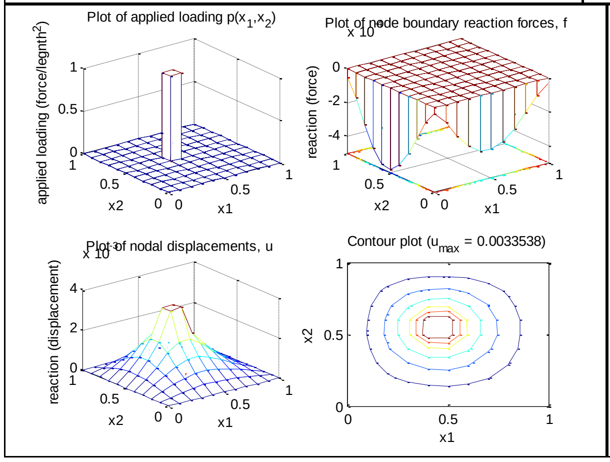


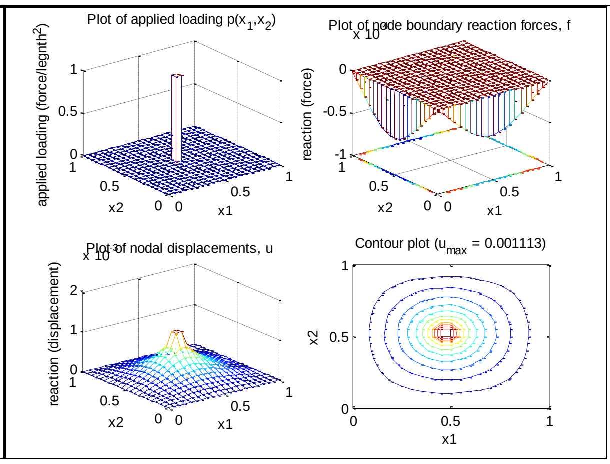
Appendix B: FEM Test Cases and results

Case i. *The following case considers a point load of p = 1 applied to the middle element of increasingly refined meshes and run to convergence (change in* u_{max} *< 1%). But notice the mesh does not converge.*

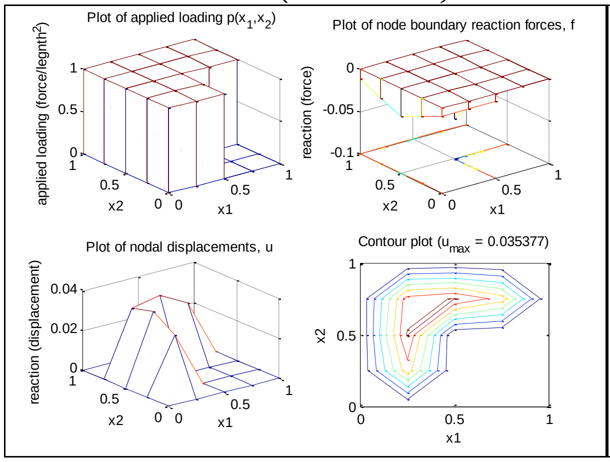


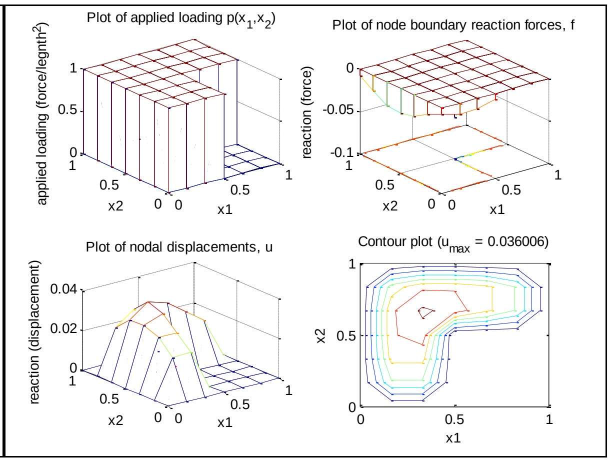


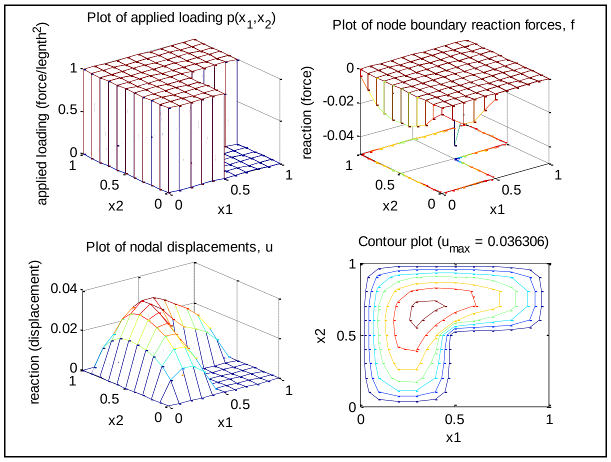




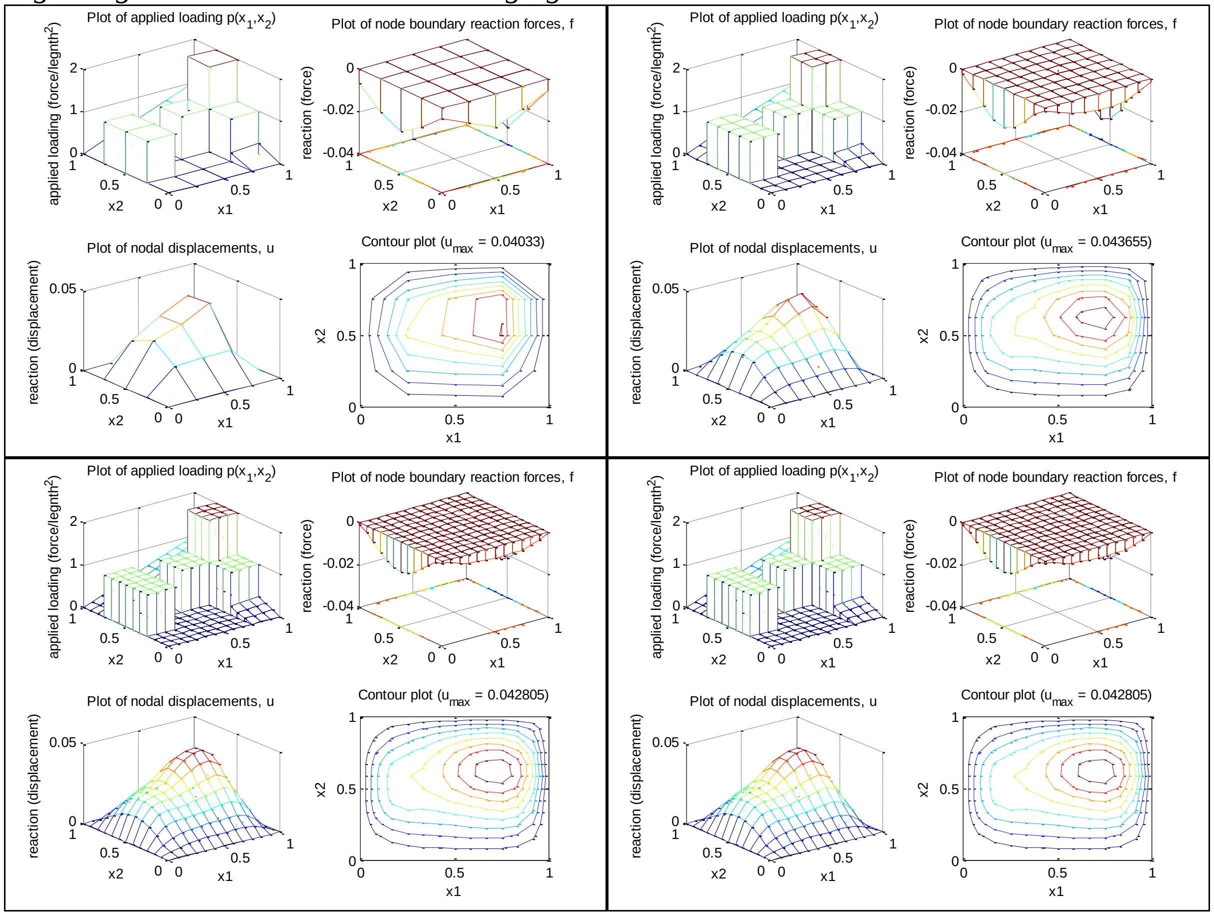
Case ii. *The following case considers an L-shaped membrane subject to uniform loading, starting with 12 elements and run to convergence (change in u_{max} < 1%). This case converges on the third iteration (75 elements).*



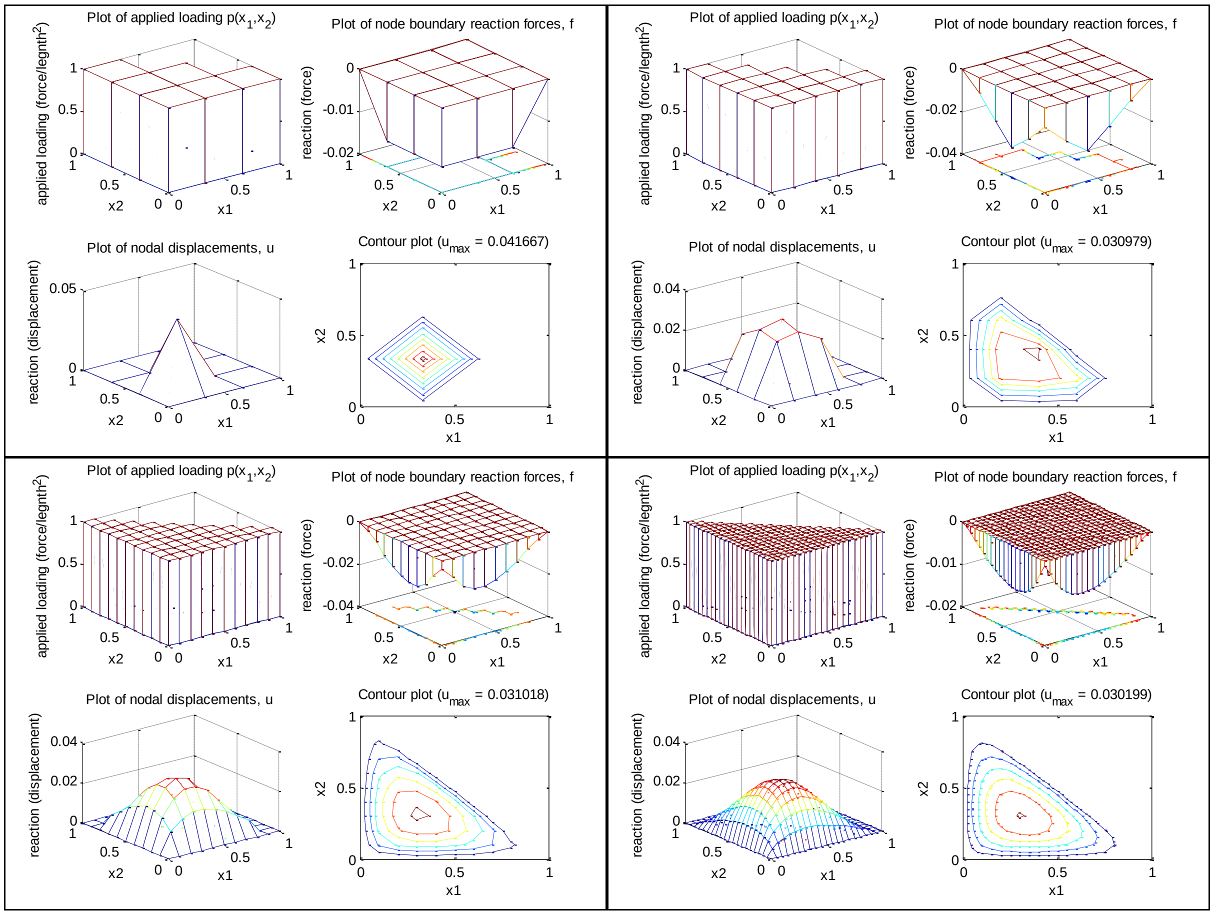




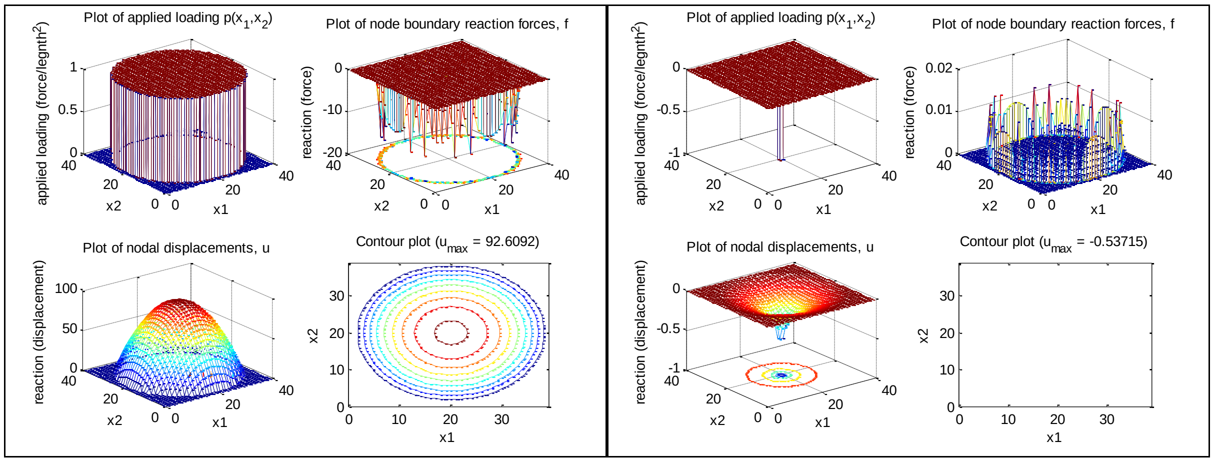
Case iii. *The following case considers a square shape subject to discontinuous loading, beginning with 16 elements and converging at 400 elements:*



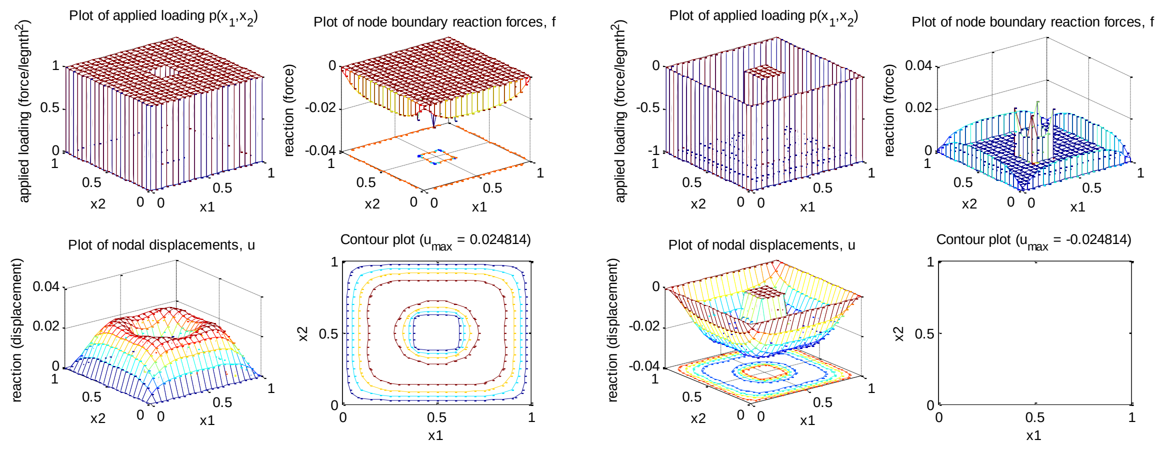
Case iv. *The following case considers a triangular membrane subject to a uniform load, starting with 6 elements and running to 210 elements.*



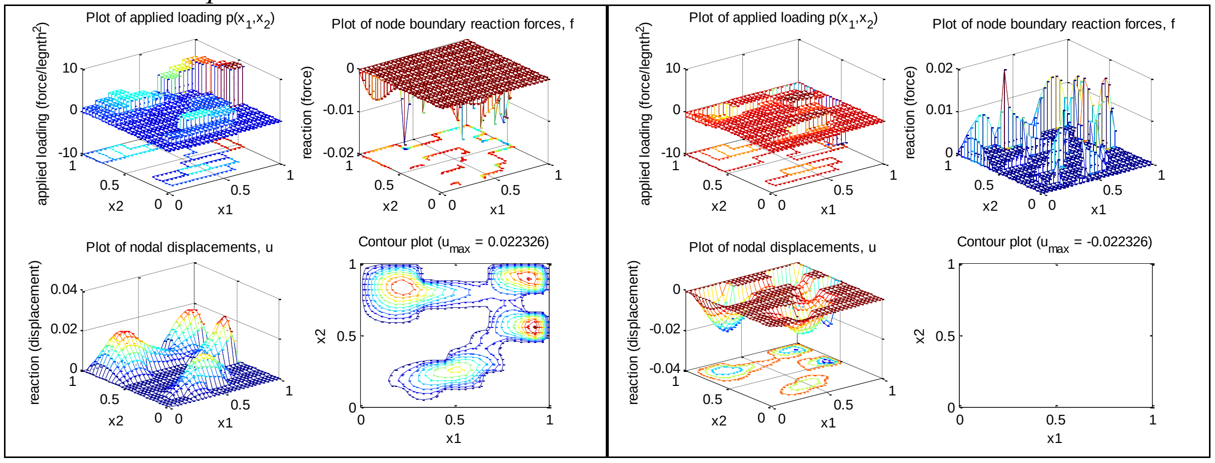
Case v. *The following case considers a circular shaped membrane of 1184 elements, first for a uniformly distributed load of p = -1 and then for a point load of p = -1 applied directly in the middle.*



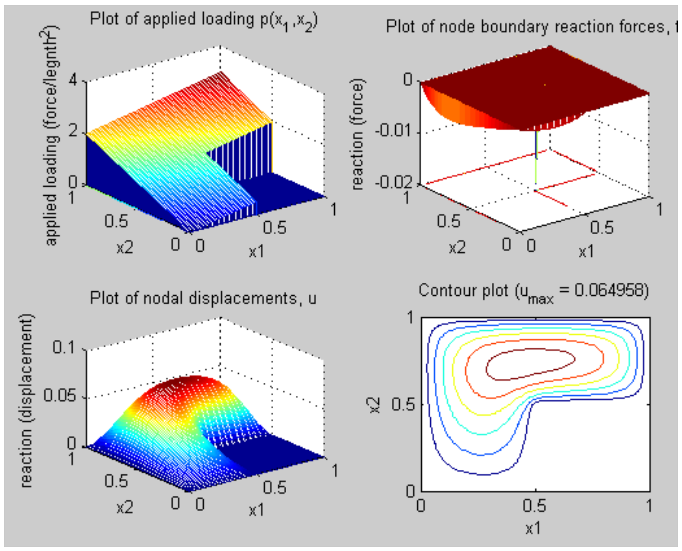
Case vi. *The following case considers a uniform load applied over a square membrane with a hole in the middle. The case is run twic – once with p, the second time with p*(-1) – to better* *visualize the membrane's response as well as the boundary reactions. This case was run with 384 elements:*



Case vii. *The following case considers a discontinuous load on a customized membrane shape. This case is run twice – one time with p, the second time with p*(-1) – to better visualize the membrane's response. This case was run with 441 elements:*



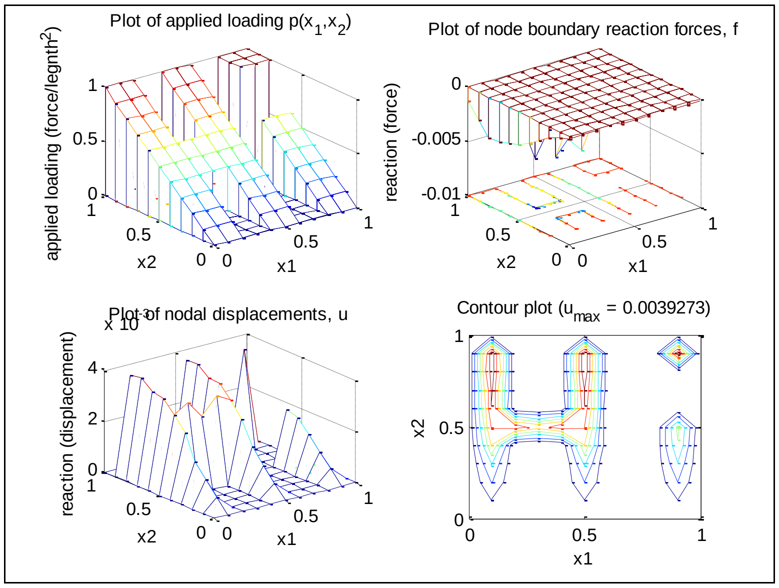
Case viii. *The following case attempts to stretch the limits of Matlab by running a linearly varying load distribution $p(x_1,x_2) = x_1+2x_2$ for an L-shape membrane with 4800 elements:*



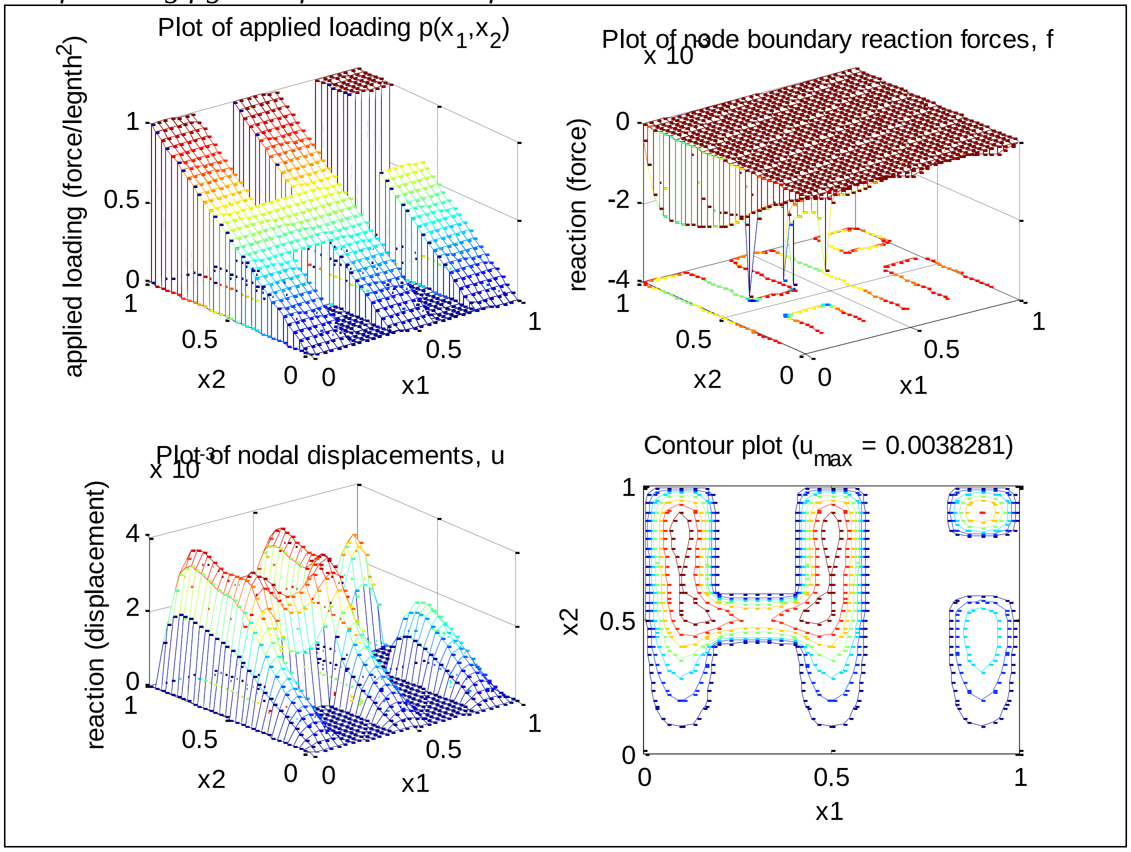
Appendix C: Custom membrane shape example, with outputs

The following example shows the definition of a custom membrane shape, a custom loading shape, the resulting MATLAB output, and finally the resulting plot. The mesh is then refined for a smoother, more accurate plot.

```
membranesubdivisions =
     2
membraneshape =
     1     0     1     0     1
     1     0     1     0     0
     1     1     1     0     1
     1     0     1     0     1
     1     0     1     0     1
loadshape =
[ x2,  0, x2,  0,  1]
[ x2,  0, x2,  0,  0]
[ x2, x2, x2,  0, x2]
[ x2,  0, x2,  0, x2]
[ x2,  0, x2,  0, x2]

Analysis on a 10 x 10 grid (60 elements):
meshsize =
    10
elemL1 =
    0.1000
elemL2 =
    0.1000

Finite element mesh, with node & element labels (nodalmatrix, elementmatrix):

        x2
        ^
        |
 1.00  91------92------93       0      94------95------96       0      97------98------99
        |       |       |               |       |       |               |       |       |
        |  55   |  56   |               |  57   |  58   |               |  59   |  60   |
        |       |       |               |       |       |               |       |       |
 0.90  85------25------86       0      87------26------88       0      89------27------90
        |       |       |               |       |       |               |       |       |
        |  49   |  50   |               |  51   |  52   |               |  53   |  54   |
        |       |       |               |       |       |               |       |       |
 0.80  78------23------79       0      80------24------81       0      82------83------84
        |       |       |               |       |       |
        |  45   |  46   |               |  47   |  48   |
        |       |       |               |       |       |
 0.70  74------21------75       0      76------22------77       0       0       0       0
        |       |       |               |       |       |
        |  41   |  42   |               |  43   |  44   |
        |       |       |               |       |       |
 0.60  66------19------67------68------69------20------70       0      71------72------73
        |       |       |       |       |       |       |               |       |       |
        |  33   |  34   |  35   |  36   |  37   |  38   |               |  39   |  40   |
        |       |       |       |       |       |       |               |       |       |
 0.50  62------13------14------15------16------17------63       0      64------18------65
        |       |       |       |       |       |       |               |       |       |
        |  25   |  26   |  27   |  28   |  29   |  30   |               |  31   |  32   |
        |       |       |       |       |       |       |               |       |       |
 0.40  55------10------56------57------58------11------59       0      60------12------61
        |       |       |               |       |       |               |       |       |
        |  19   |  20   |               |  21   |  22   |               |  23   |  24   |
        |       |       |               |       |       |               |       |       |
 0.30  49------ 7------50       0      51------ 8------52       0      53------ 9------54
        |       |       |               |       |       |               |       |       |
        |  13   |  14   |               |  15   |  16   |               |  17   |  18   |
        |       |       |               |       |       |               |       |       |
 0.20  43------ 4------44       0      45------ 5------46       0      47------ 6------48
        |       |       |               |       |       |               |       |       |
        |   7   |   8   |               |   9   |  10   |               |  11   |  12   |
        |       |       |               |       |       |               |       |       |
 0.10  37------ 1------38       0      39------ 2------40       0      41------ 3------42
        |       |       |               |       |       |               |       |       |
        |   1   |   2   |               |   3   |   4   |               |   5   |   6   |
        |       |       |               |       |       |               |       |       |
 0.00  28------29------30------ 0------31------32------33------ 0------34------35------36 --> x1

      0.00    0.10    0.20    0.30    0.40    0.50    0.60    0.70    0.80    0.90    1.00

Node coordinates (nodalcoords):

| node| (   x1,   x2) |
|   1 | ( 0.10, 0.10) |
|   2 | ( 0.50, 0.10) |
|   3 | ( 0.90, 0.10) |
|   4 | ( 0.10, 0.20) |
|   5 | ( 0.50, 0.20) |
|   6 | ( 0.90, 0.20) |
|   7 | ( 0.10, 0.30) |
|   8 | ( 0.50, 0.30) |
|   9 | ( 0.90, 0.30) |
|  10 | ( 0.10, 0.40) |
|  11 | ( 0.50, 0.40) |
|  12 | ( 0.90, 0.40) |
|  13 | ( 0.10, 0.50) |
|  14 | ( 0.20, 0.50) |
|  15 | ( 0.30, 0.50) |
|  16 | ( 0.40, 0.50) |
|  17 | ( 0.50, 0.50) |
|  18 | ( 0.90, 0.50) |
|  19 | ( 0.10, 0.60) |
|  20 | ( 0.50, 0.60) |
|  21 | ( 0.10, 0.70) |
|  22 | ( 0.50, 0.70) |
|  23 | ( 0.10, 0.80) |
|  24 | ( 0.50, 0.80) |
|  25 | ( 0.10, 0.90) |
|  26 | ( 0.50, 0.90) |
|  27 | ( 0.90, 0.90) |
|  28 | ( 0.00, 0.00) |
|  29 | ( 0.10, 0.00) |
|  30 | ( 0.20, 0.00) |
|  31 | ( 0.40, 0.00) |
|  32 | ( 0.50, 0.00) |
|  33 | ( 0.60, 0.00) |
|  34 | ( 0.80, 0.00) |
|  35 | ( 0.90, 0.00) |
|  36 | ( 1.00, 0.00) |
|  37 | ( 0.00, 0.10) |
|  38 | ( 0.20, 0.10) |
|  39 | ( 0.40, 0.10) |
|  40 | ( 0.60, 0.10) |
|  41 | ( 0.80, 0.10) |
|  42 | ( 1.00, 0.10) |
|  43 | ( 0.00, 0.20) |
|  44 | ( 0.20, 0.20) |
|  45 | ( 0.40, 0.20) |
|  46 | ( 0.60, 0.20) |
|  47 | ( 0.80, 0.20) |
|  48 | ( 1.00, 0.20) |
|  49 | ( 0.00, 0.30) |
|  50 | ( 0.20, 0.30) |
|  51 | ( 0.40, 0.30) |
|  52 | ( 0.60, 0.30) |
|  53 | ( 0.80, 0.30) |
|  54 | ( 1.00, 0.30) |
|  55 | ( 0.00, 0.40) |
|  56 | ( 0.20, 0.40) |
|  57 | ( 0.30, 0.40) |
|  58 | ( 0.40, 0.40) |
|  59 | ( 0.60, 0.40) |
|  60 | ( 0.80, 0.40) |
|  61 | ( 1.00, 0.40) |
|  62 | ( 0.00, 0.50) |
|  63 | ( 0.60, 0.50) |
|  64 | ( 0.80, 0.50) |
|  65 | ( 1.00, 0.50) |
|  66 | ( 0.00, 0.60) |
|  67 | ( 0.20, 0.60) |
|  68 | ( 0.30, 0.60) |
|  69 | ( 0.40, 0.60) |
|  70 | ( 0.60, 0.60) |
|  71 | ( 0.80, 0.60) |
|  72 | ( 0.90, 0.60) |
|  73 | ( 1.00, 0.60) |
|  74 | ( 0.00, 0.70) |
|  75 | ( 0.20, 0.70) |
|  76 | ( 0.40, 0.70) |
|  77 | ( 0.60, 0.70) |
|  78 | ( 0.00, 0.80) |
|  79 | ( 0.20, 0.80) |
|  80 | ( 0.40, 0.80) |
|  81 | ( 0.60, 0.80) |
|  82 | ( 0.80, 0.80) |
|  83 | ( 0.90, 0.80) |
|  84 | ( 1.00, 0.80) |
|  85 | ( 0.00, 0.90) |
|  86 | ( 0.20, 0.90) |
|  87 | ( 0.40, 0.90) |
|  88 | ( 0.60, 0.90) |
|  89 | ( 0.80, 0.90) |
|  90 | ( 1.00, 0.90) |
|  91 | ( 0.00, 1.00) |
|  92 | ( 0.10, 1.00) |
|  93 | ( 0.20, 1.00) |
|  94 | ( 0.40, 1.00) |
|  95 | ( 0.50, 1.00) |
|  96 | ( 0.60, 1.00) |
|  97 | ( 0.80, 1.00) |
|  98 | ( 0.90, 1.00) |
|  99 | ( 1.00, 1.00) |

Element connectivity matrix (elemnodes):
|       |      local node         |
|element|   a |   b |   c |   d |
|      1 |  28 |  29 |  37 |   1 |
|      2 |  29 |  30 |   1 |  38 |
|      3 |  31 |  32 |  39 |   2 |
|      4 |  32 |  33 |   2 |  40 |
|      5 |  34 |  35 |  41 |   3 |
|      6 |  35 |  36 |   3 |  42 |
|      7 |  37 |   1 |  43 |   4 |
|      8 |   1 |  38 |   4 |  44 |
|      9 |  39 |   2 |  45 |   5 |
|     10 |   2 |  40 |   5 |  46 |
|     11 |  41 |   3 |  47 |   6 |
|     12 |   3 |  42 |   6 |  48 |
|     13 |  43 |   4 |  49 |   7 |
|     14 |   4 |  44 |   7 |  50 |
|     15 |  45 |   5 |  51 |   8 |
|     16 |   5 |  46 |   8 |  52 |
|     17 |  47 |   6 |  53 |   9 |
|     18 |   6 |  48 |   9 |  54 |
|     19 |  49 |   7 |  55 |  10 |
|     20 |   7 |  50 |  10 |  56 |
|     21 |  51 |   8 |  58 |  11 |
|     22 |   8 |  52 |  11 |  59 |
|     23 |  53 |   9 |  60 |  12 |
|     24 |   9 |  54 |  12 |  61 |
|     25 |  55 |  10 |  62 |  13 |
|     26 |  10 |  56 |  13 |  14 |
|     27 |  56 |  57 |  14 |  15 |
|     28 |  57 |  58 |  15 |  16 |
|     29 |  58 |  11 |  16 |  17 |
|     30 |  11 |  59 |  17 |  63 |
|     31 |  60 |  12 |  64 |  18 |
|     32 |  12 |  61 |  18 |  65 |
|     33 |  62 |  13 |  66 |  19 |
|     34 |  13 |  14 |  19 |  67 |
|     35 |  14 |  15 |  67 |  68 |
|     36 |  15 |  16 |  68 |  69 |
|     37 |  16 |  17 |  69 |  20 |
|     38 |  17 |  63 |  20 |  70 |
|     39 |  64 |  18 |  71 |  72 |
|     40 |  18 |  65 |  72 |  73 |
|     41 |  66 |  19 |  74 |  21 |
|     42 |  19 |  67 |  21 |  75 |
|     43 |  69 |  20 |  76 |  22 |
|     44 |  20 |  70 |  22 |  77 |
|     45 |  74 |  21 |  78 |  23 |
|     46 |  21 |  75 |  23 |  79 |
|     47 |  76 |  22 |  80 |  24 |
|     48 |  22 |  77 |  24 |  81 |
|     49 |  78 |  23 |  85 |  25 |
|     50 |  23 |  79 |  25 |  86 |
|     51 |  80 |  24 |  87 |  26 |
|     52 |  24 |  81 |  26 |  88 |
|     53 |  82 |  83 |  89 |  27 |
|     54 |  83 |  84 |  27 |  90 |
|     55 |  85 |  25 |  91 |  92 |
|     56 |  25 |  86 |  92 |  93 |
|     57 |  87 |  26 |  94 |  95 |
|     58 |  26 |  88 |  95 |  96 |
|     59 |  89 |  27 |  97 |  98 |
|     60 |  27 |  90 |  98 |  99 |

Maximum displacement = 0.00392729
```



The following figure refines the mesh for 540 elements:



Appendix D: MATLAB Code

See attached