Summary of Validation Testing for a Parallel Mesh Generation Toolbox

Wen Yu

September 2008

Computing and Software

McMaster University

Wen Yu – McMaster University – Computing and Software

Contents

1 Introduction **2**
- 1.1 Purpose of the Document 2
- 1.2 Scope of the Testing 2
- 1.3 Organization of the Document 2

2 Testing PMGT **2**
- 2.1 Test Cases 3
  - 2.1.1 Automated Correctness Validation Tests Requirements 3
  - 2.1.2 Visual Correctness Validation Tests Requirements . . . 4
  - 2.1.3 Test Cases 4
- 2.2 Traceability Matrix for SRS 6
- 2.3 Traceability Matrix for MG 8

3 Results and Analysis **8**
- 3.1 Testing Results 9
- 3.2 Analysis 12

1 Introduction

This section gives an overview of the Testing Summary for a Parallel Mesh Generation Toolbox (PMGT). First, the purpose of the document is provided. Second, the scope of the testing is identified. Third, the organization of the document is summarized.

1.1 Purpose of the Document

This document specifies validation tests for a PMGT. The results of the tests and analysis are also provided. The intended audience is testers who are going to test the system and developers who are going to maintain the software. Note that test document is dynamic in the sense that it should be updated when the development of the system proceeds.

1.2 Scope of the Testing

In general, the purpose of testing is to help produce quality software. Due to limits on the time available for testing, the scope of the testing of PMGT is restricted to test the most important test factors. Like other scientific computing software, correctness and efficiency are considered to be the two most important test factors for PMGT. For efficiency testing, the focus is on execution time rather than on storage.

1.3 Organization of the Document

Section 1 (this section) is an introduction to the report. Section 2 shows what is going to be tested and the coverage of the testing, with respect to the software requirements and the software design. Section 3 gives the result of the testing and the analysis.

2 Testing PMGT

Test cases are listed in Section 2.1. The detailed information for these test cases can be found in Section 3. The traceability matrix in Section 2.2 shows the association between test cases and the functional and nonfunctional requirements that are specified in the Software Requirements Specification (SRS) document. Similarly, a traceability matrix for test cases and the leaf modules as introduced in the Module Guide (MG) as shown in Section 2.3. Tracking these relations is useful for developing and maintaining the software.

2.1 Test Cases

The correctness validation test is designed for verifying the functional requirements RefiningMesh (F1), CoarseningMesh (F2), ElmShape (F5), and Conformal (F7). Other requirements for correctness are trivial and are satisfied obviously. For example, since the vertices are stored in an array, the OutVertexOrder (F15) requirement is met by outputting the vertices in the order as the order of them in the array. The tests are against above requirements are automated. The automated validation tests requirements (ACVTRs) are listed in Section 2.1.1. Since the output mesh also can also be displayed on screen, it can be checked manually. The visual correctness validation tests requirements (VCVTRs) are listed in Section 2.1.2. The test cases are in Section 2.1.3.

2.1.1 Automated Correctness Validation Tests Requirements

A list of ACVTRs follows. All test cases should pass these tests. Some test cases relate to data definitions defined in the SRS. In these cases the related data definition defined is shown as Dx, where x is the number of the associated data definition given in the SRS.

- The area of each element is greater than zero (referring to **D5**).
- The boundary of the mesh is closed. (referring to **D15**).
- The mesh is conformal (referring to **D16**).
- The intersection of any two elements is empty (referring to **D17**).
- The input mesh and output mesh *CoveringUp* each other (referring to **D19**).
- The length of each edge is greater than zero. (This is required by the definition of a mesh, which is defined in the SRS.)
- The vertices of each element are listed in a counterclockwise order. (The counterclockwise order of the vertices for each element is not necessary for implementing PMGT. However, it is adopted by most meshing and FEA software. PMGT uses this convention.)
- The output mesh conforms to the Euler Equation. (This requirement is not documented in the SRS. However, any mesh should implicitly satisfy the equation $nc + nv - ne = 1$, where nc is the number of cells, nv is the number of vertices, and ne is the number of edges.)

2.1.2 Visual Correctness Validation Tests Requirements

The output meshes should also be visually checked to ensure that the following VCVTRs are met.

- No vertex is outside of the input domain.
- No vertex is inside of a cell.
- No dangling points or edges are present.
- All cells are connected.
- The mesh is conformal.

Some of the VCVTRs overlap with the ACVTRs. This redundancy provides increased confidence in case one testing method fails to catch an error.

2.1.3 Test Cases

The test cases developed involve testing meshes against the above requirements. In each test case, except the last one, the input mesh is refined and then coarsened. Two algorithms for refining are used. One algorithm is called *Split*. It splits one cell into three by adding a point in the centroid of the triangle and connecting the added point to the three original vertices. The other algorithm is simply call *Refine*. It refines the original mesh by longest edge bisection.

The name of each test case includes three parts. For example, test case $AxxCBN$ means that the test uses Axx algorithm for refining, where Axx equals $Split$ or $Refine$. The letter C indicates that coarsening is performed. If the C is missing, the input mesh is not coarsened. B is the number of refinements before coarsening. If B is S, the mesh is refined once and then coarsen once. If the B is M, the mesh is refined multiple time before coarsening. N is a number. If the N is omitted, it means only one of this kind of test performed. Otherwise the same test procedures is used several times on different input meshes. The reason for using the same procedure is that the topology of the output meshes may differ for different input meshes.

- Test Case *SplitCS* (TC1): This test case tests the correctness of PMGT. The input mesh is shown in Figure 3. The refining and coarsening criterion is that the cells intersected with the vertical line, $x = 0.6$, are Split once, then the cells of the new mesh that intersect with the vertical line are coarsened once. When the splitting and coarsening is done, the

vertical line is moved to the right one unit ($x = x + 1.0$), and another Splitting and coarsening is performed. This procedure is repeated until no cells intersect with the vertical line.

- Test Case *RefineCS1* (TC2): This test case tests the correctness of PMGT. The input mesh is the same as TC1, which is shown in Figure 3. There is a vertical line at $x = 0.6$. The refining and coarsening criterion is that the cells that intersect with the vertical line are refined once, then the cells of the new mesh that intersect with the vertical line are coarsened once. When the refining and coarsening are done, the vertical line is moved to the right one unit, and another refining and coarsening is performed. This procedure is repeated until no cells intersect with the vertical line.

- Test Case *RefineCS2* (TC3): This test case tests the correctness of PMGT. The refining and coarsening criterion, vertical line function, and the test procedure are the same as test case TC2. However, the input mesh is different. The input mesh is showed in Figure 4.

- Test Case *RefineCM* (TC4): This test case tests the correctness of PMGT. The input mesh is shown in Figure 5. There is a vertical line at $x = 0.5$. The refining and coarsening criterion is the size of the cells. The size of the cell is measured by the length of the longest edge of the cell. The cells that intersect with the vertical line are refined until the criterion is met. When the refining is done, the vertical line is moved to the right 0.6 unit ($x = x + 0.6$), and another refinement is performed. After five refinements are done, the cells to be left of the vertical line by up to 2 units are coarsened, until the coarsening criterion is met. The refining and coarsening are stopped when the vertical line moves to a position outside of the domain.

- Test Case *RefineM* (TC5): This test case tests the correctness of PMGT. The input mesh is shown in Figure 6. There is an arc with radius of 0.7 unit going through the mesh. Cells that intersect with the arc are refined until the required number of refinements has been reached.

- Test Case *Split* (TC6): This test case tests both the correctness and speed of PMGT. The input mesh is shown in Figure 7. This test simply splits all cells of the mesh 4 times. It is done in both the serial version and the parallel version with different number of processors. The execution time of setting the cells to be refined and splitting the cells is measured.

2.2 Traceability Matrix for SRS

In the traceability matrix for software requirements, if a test case tests the functionality of a software requirement, there will be a check mark on the cell for the corresponding test case. In each row of the traceability matrix for software requirements (Table 1), if the requirement in that row defines the correctness or the speed of the software, one or more cells in this row are checked. Otherwise, all cells in the row are empty. Table 1 shows that the test cases developed in Section 2.1 assist with validating the correctness and speed of the software. The detailed information for each functional and nonfunctional requirements can be found in the SRS document. The names of the requirements and their corresponding numbers are listed below for convenience.

F1: RefiningMesh

F2: CoarseningMesh

F3: RefiningOrCoarsening

F4: MeshType

F5: ElmShape

F6: DomainDimension

F7: Conformal

F8: InputDefinition

F9: RCInstruction

F10: OutputStorage

F11: VertexUniqueID

F12: ElmUniqueID

F13: ElmTopology

F14: OutElmOrder

F15: OutVertexOrder

F16: Help

N1: Performance

N2: Precision

N3: Exception

N4: Portability

N5: LookAndFeel

N6: Usability

N7: Maintainability

	TC1	TC2	TC3	TC4	TC5	TC6
F1	✓	✓	✓	✓	✓	✓
F2	✓	✓	✓	✓		
F3	✓	✓	✓	✓	✓	✓
F4	✓	✓	✓	✓	✓	
F5	✓	✓	✓	✓	✓	
F6	✓	✓	✓	✓	✓	
F7	✓	✓	✓	✓	✓	
F8	✓	✓	✓	✓	✓	✓
F9	✓	✓	✓	✓	✓	✓
F10	✓	✓	✓	✓	✓	
F11	✓	✓	✓	✓	✓	
F12	✓	✓	✓	✓	✓	
F13	✓	✓	✓	✓	✓	
F14	✓	✓	✓	✓	✓	
F15	✓	✓	✓	✓	✓	
F16						
N1						✓
N2						
N3						
N4						
N5						
N6						
N7						

Table 1: Traceability Matrix: Test Cases and Requirements

2.3 Traceability Matrix for MG

Similar to Section 2.2, the traceability matrix for modules (Table 2) shows that the test cases validate the modules that are associated with correctness and speed. The names of modules appear in Table 2 are listed below. The detailed information for each module can be found in the MG document.

M1: Virtual Memory Module

M2: File Read/Write Module

M3: Keyboard Input Module

M4: Screen Output Module

M5: Input Format Module

M6: Output Format Module

M7: Service Module

M8: Vertex Module

M9: Edge Module

M10: Cell Module

M11: Mesh Module

M12: Refining Module

M13: Coarsening Module

3 Results and Analysis

The results of the test cases defined in Section 2.1.3 are listed in Section 3.1. The analysis, including charts that compare the execution time of the parallel version to the serial version are provided in Section 3.2.

	TC1	TC2	TC3	TC4	TC5	TC6
M1	✓	✓	✓	✓	✓	✓
M2	✓	✓	✓	✓	✓	✓
M3	✓	✓	✓	✓	✓	✓
M4	✓	✓	✓	✓	✓	✓
M5	✓	✓	✓	✓	✓	✓
M6	✓	✓	✓	✓	✓	✓
M7	✓	✓	✓	✓	✓	
M8	✓	✓	✓	✓	✓	✓
M9	✓	✓	✓	✓	✓	✓
M10	✓	✓	✓	✓	✓	✓
M11	✓	✓	✓	✓	✓	✓
M12	✓	✓	✓	✓	✓	✓
M13	✓	✓	✓	✓		

Table 2: Traceability Matrix: Test Cases and Modules

3.1 Testing Results

The following tables list the testing results of each test case. The field *Test Case Number* and *Test Case Name* list the number and the name of each test case. The *Input field* gives the number of the figure that is the input for that test case, or a description of the input mesh. The *Expected Output* describes the requirements of the output mesh. The *Actual Output* gives the result of the test. The *Selected Output Mesh* field should give the output meshes. However, there are too many intermediate mesh to display, and displaying only the final mesh is too simple to illustrate the feature of the test case. Selected intermediate meshes and final mesh are included in the *Actual Output* field. The *Result* field indicates whether the test is passed or failed.

Test Case Number	TC1
Test Case Name	SplitCS
Input	Figure 3
Expected Output	ACVTRs and VCVTRs listed in Section 2 are met
Actual Output	Summary of the correctness test: 15 tests are performed. 15 tests succeed. 0 tests fail.
Selected Output Mesh	Figure 8, 9, 10
Result	Passed

Test Case Number	TC2
Test Case Name	RefineCS1
Input	Figure 3
Expected Output	ACVTRs and VCVTRs listed in Section 2 are met
Actual Output	Summary of the correctness test: 15 tests are performed. 15 tests succeed. 0 tests fail.
Selected Output Mesh	Figure 11, 12, 13
Result	Passed

Test Case Number	TC3
Test Case Name	RefineCS2
Input	Figure 4
Expected Output	ACVTRs and VCVTRs listed in Section 2 are met
Actual Output	Summary of the correctness test: 15 tests are performed. 15 tests succeed. 0 tests fail.
Selected Output Mesh	Figure 14, 15, 16
Result	Passed

Test Case Number	TC4
Test Case Name	RefineCM
Input	Figure 5
Expected Output	ACVTRs and VCVTRs listed in Section 2 are met
Actual Output	Summary of the correctness test: 15 tests are performed. 15 tests succeed. 0 tests fail.
Selected Output Mesh	Figure 17, 18, 19, 20
Result	Passed

Test Case Number	TC5
Test Case Name	RefineM
Input	Figure 6
Expected Output	ACVTRs and VCVTRs listed in Section 2 are met
Actual Output	Summary of the correctness test: 15 tests are performed. 15 tests succeed. 0 tests fail.
Selected Output Mesh	Figure 21, 22, 23
Result	Passed

Test Case Number	TC6
Test Case Name	SplitM
Input	Figure 6
Expected Output	ACVTRs and VCVTRs listed in Section 2 are met Execution time increases as the number of cells increases. Execution time decreases as the number of processors increases.
Actual Output	Execution time as indicated in Figure 1
Selected Output Mesh	The mesh is too dense to be shown.
Result	Passed

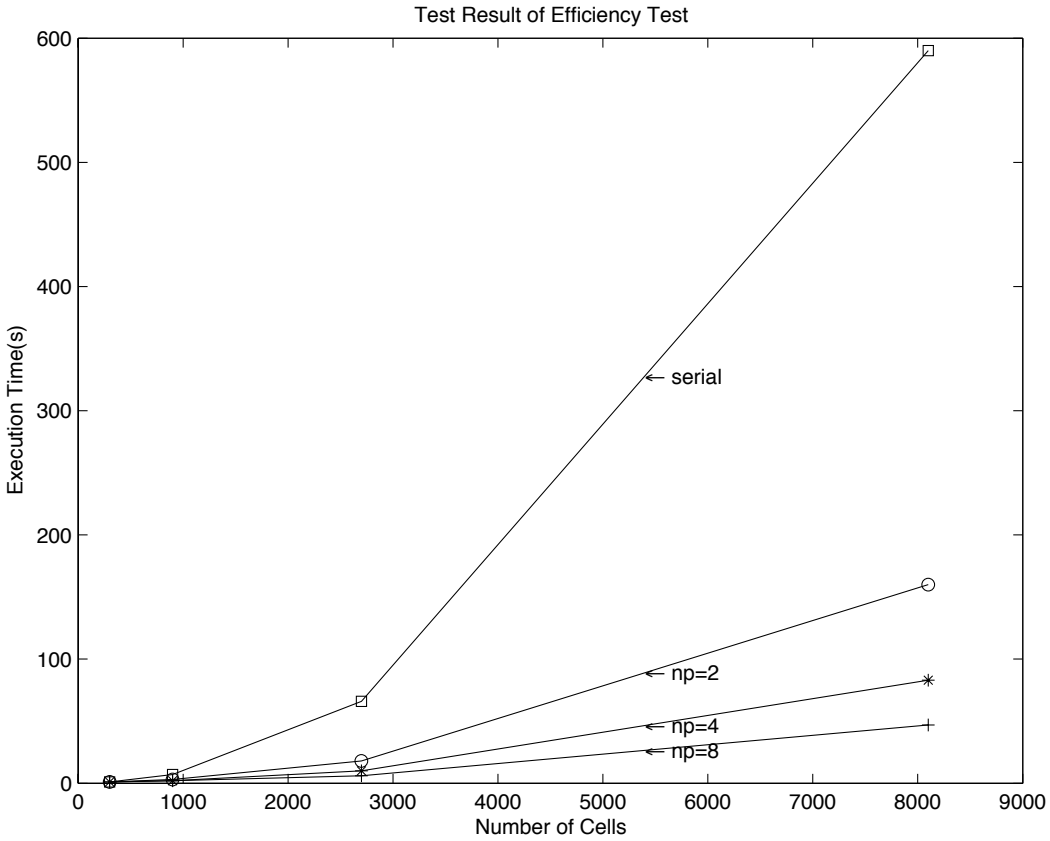

Figure 1: Output of TC6

3.2 Analysis

All of the test cases conform to the ACVTRs and VCVTRs listed in Section 2. The test result of TC6 show that when the number of cells increased, the execution time increased, and when the number of processors increased, the execution time decreased. That is, this test is passed. Figure 2 show the speedup when using different numbers of processors. The speedup is defined as

$$Speedup(n) = \frac{T_1}{T_n}$$

Where T_1 is the execution time of the serial version, and T_n is the execution time of the parallel version with n processors. In general, $Speedup(n) < n$. However, for PMGT, when the number of cells is greater than 2700, $Speedup(n) > n$, which is a super linear speedup. Since the algorithms used for the serial version and the parallel version are the same, the super linear

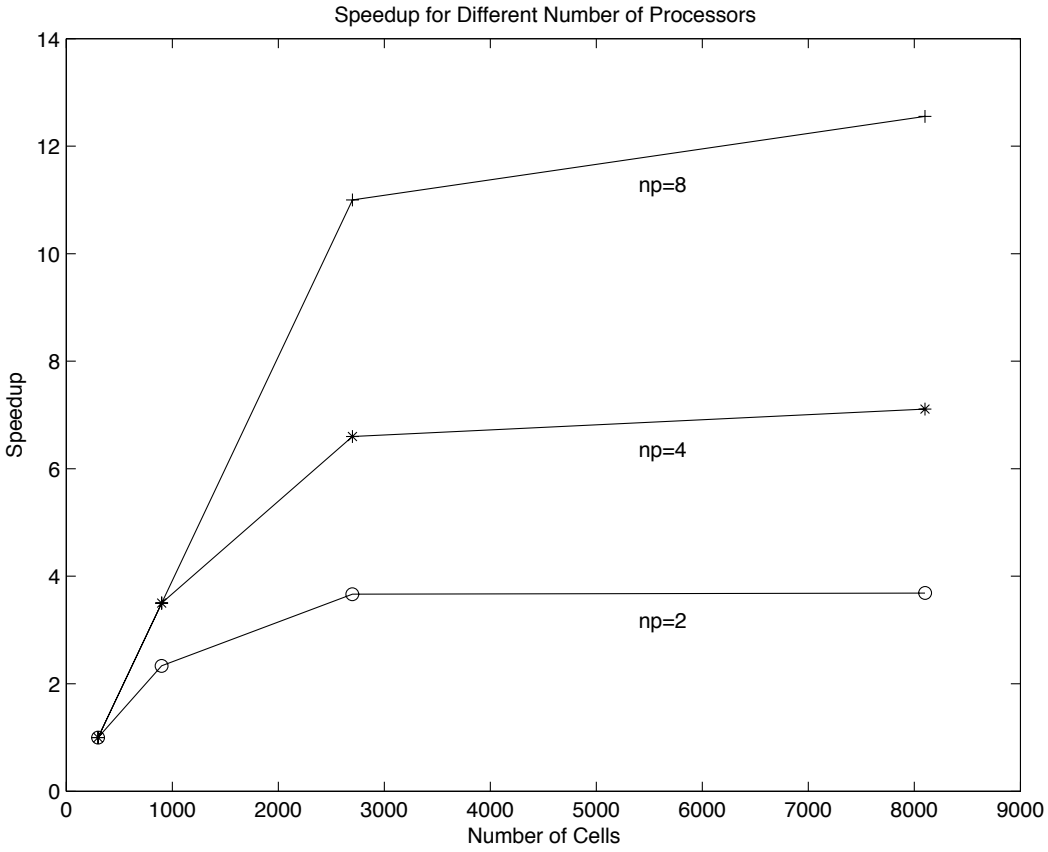

Figure 2: Speedup for Different Numbers of Processors

speedup is probably due to the cache effect. That is, when the numbers of processors increases, the size of the accumulated caches from different processors also increases. With the larger accumulated cache size, more, or even all, core data set can fit into the caches and the memory access time reduces dramatically. This may explain the extra speedup in additional to the speedup due to parallel computation.

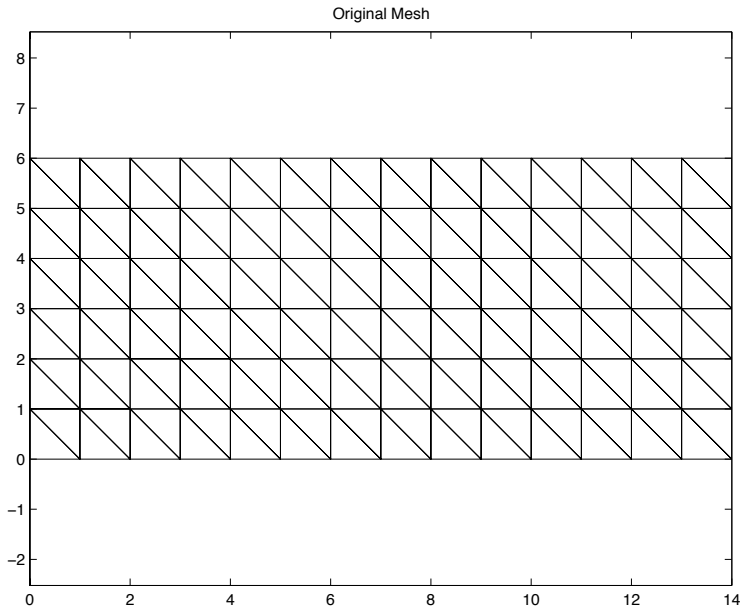

Figure 3: Input 1

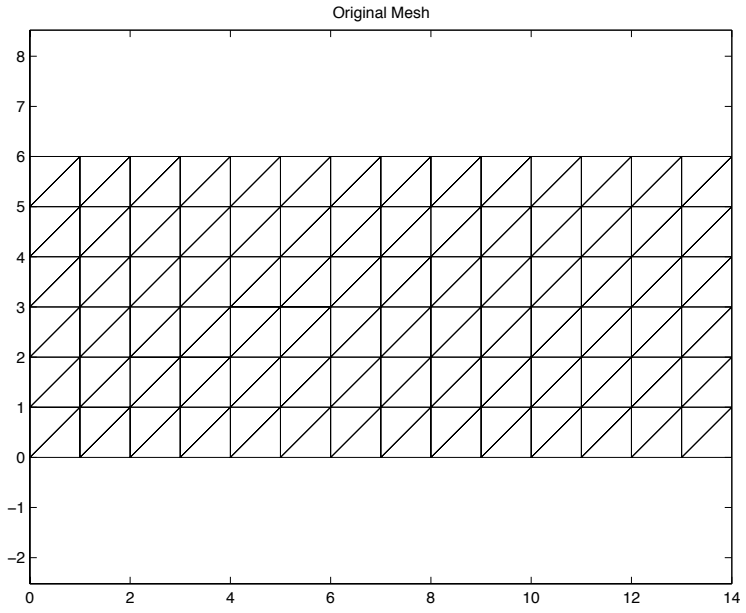

Figure 4: Input 2

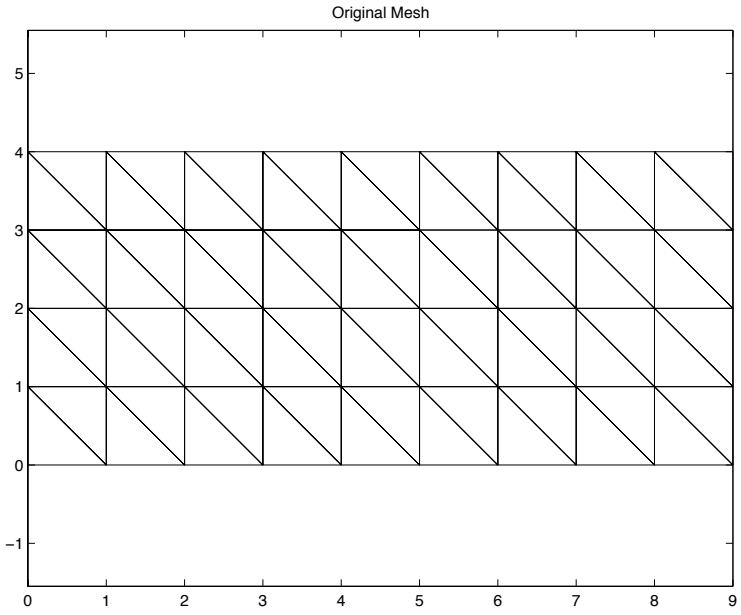

Figure 5: Input 3

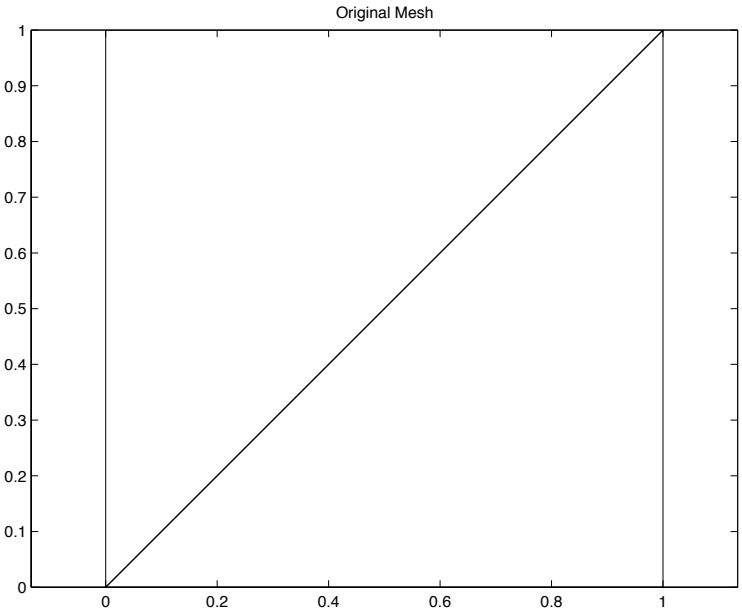

Figure 6: Input 4

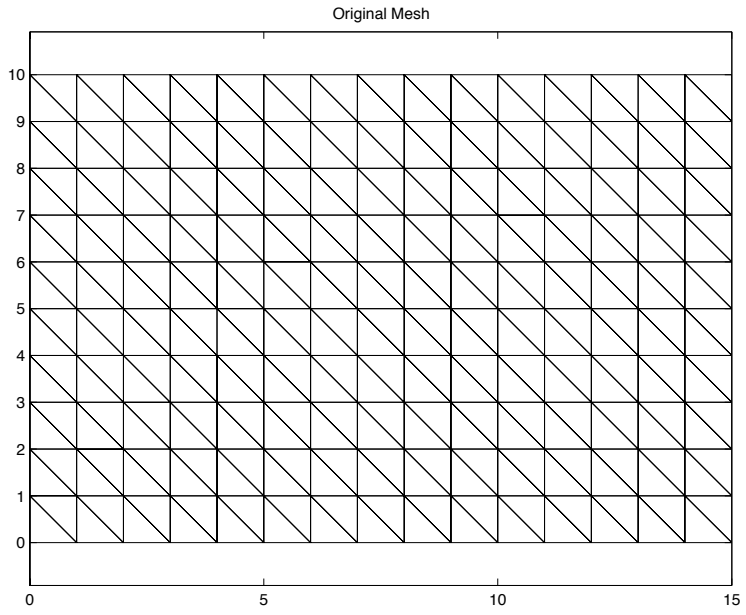

Figure 7: Input 5

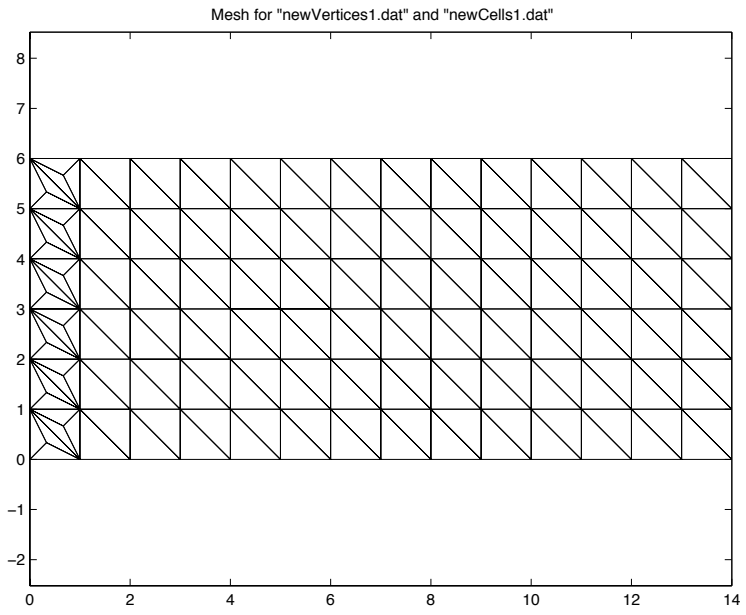

Figure 8: Output 1 of TC1

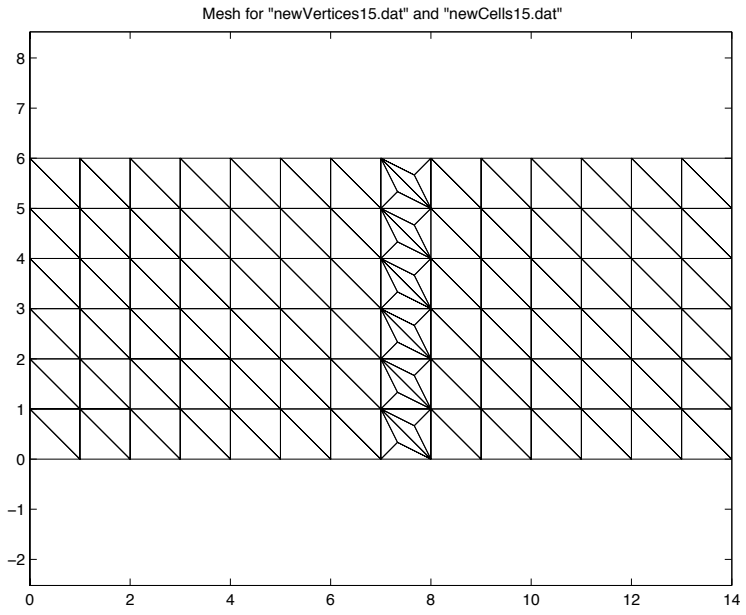

Figure 9: Output 2 of TC1

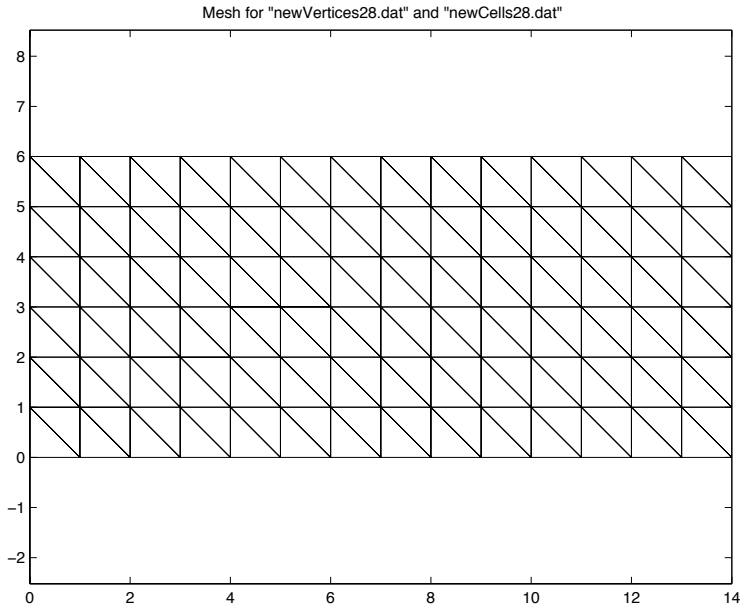

Figure 10: Output 3 of TC1

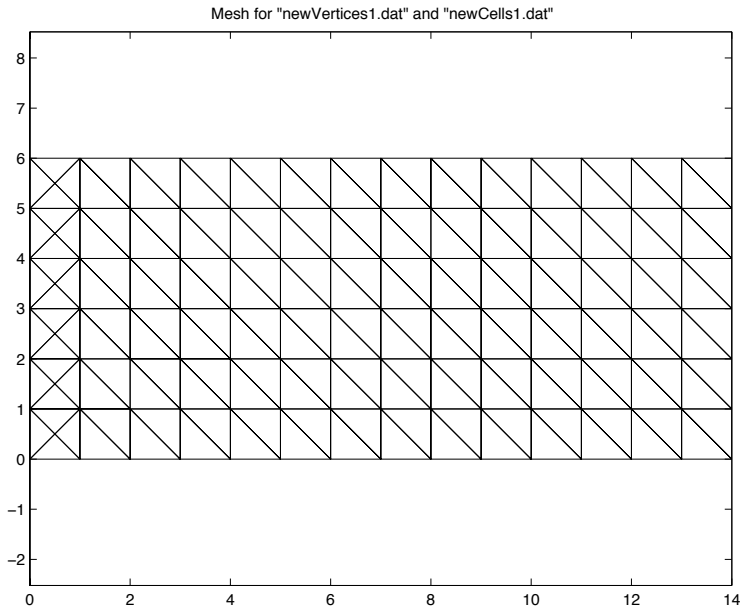

Figure 11: Output 1 of TC2

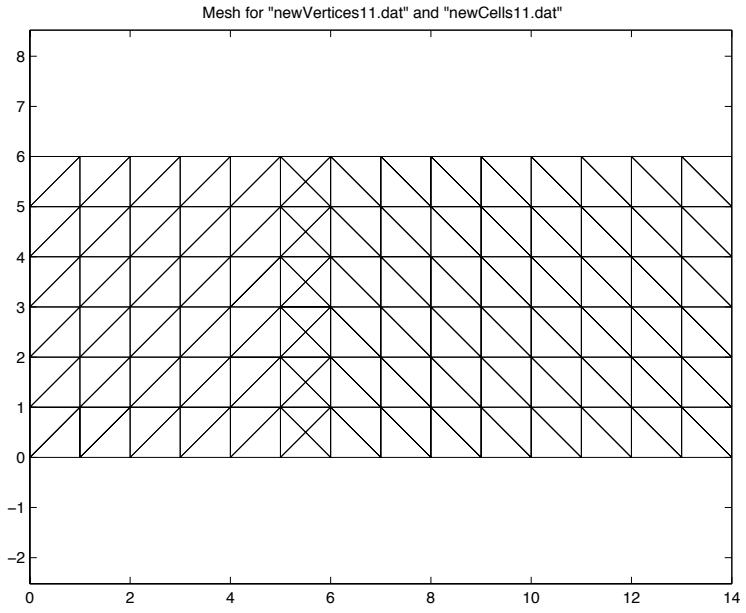

Figure 12: Output 2 of TC2

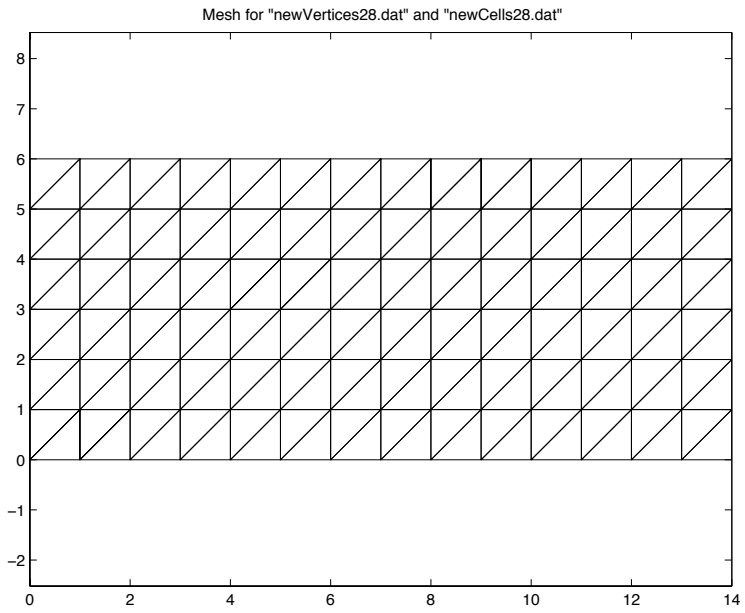

Figure 13: Output 3 of TC2

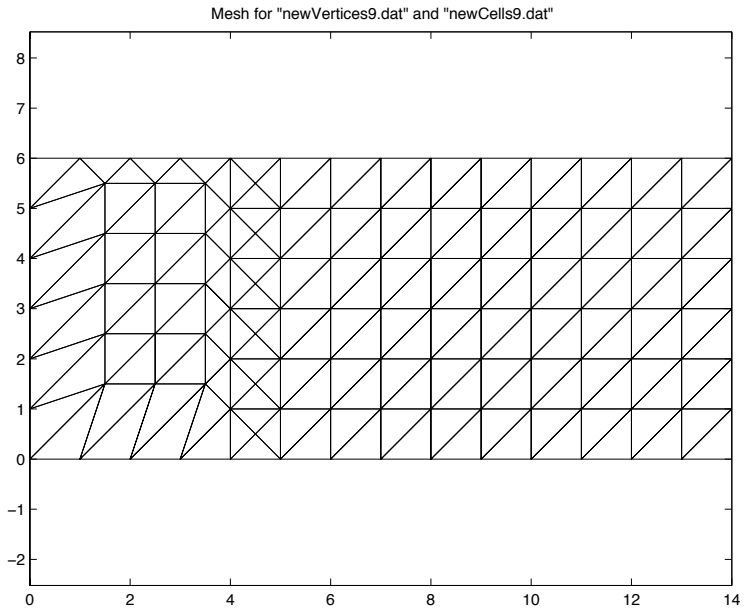

Figure 14: Output 1 of TC3

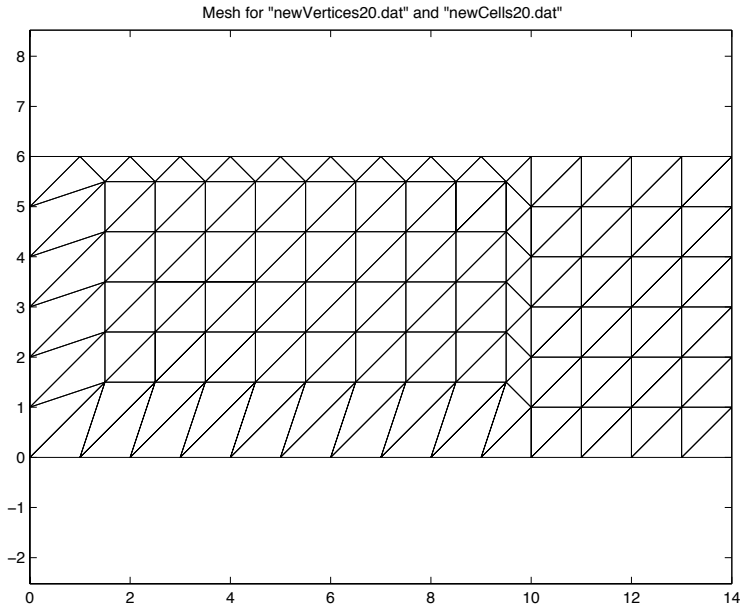

Figure 15: Output 2 of TC3

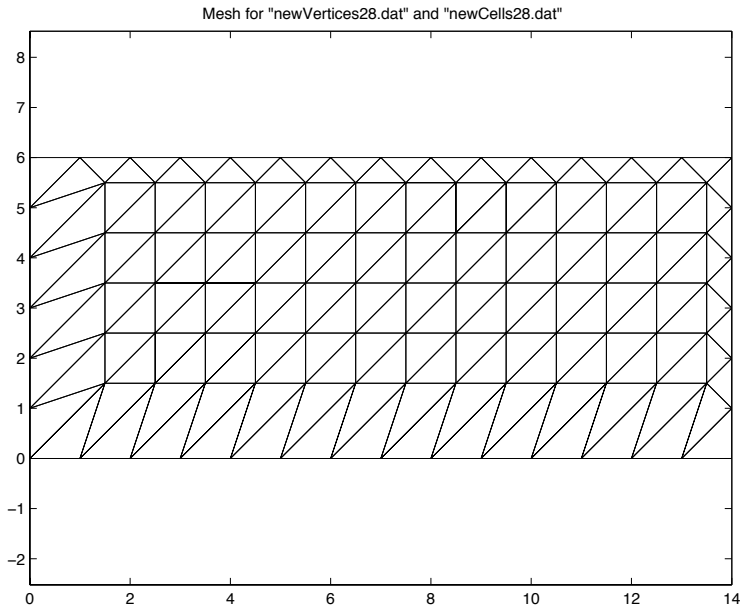

Figure 16: Output 3 of TC3

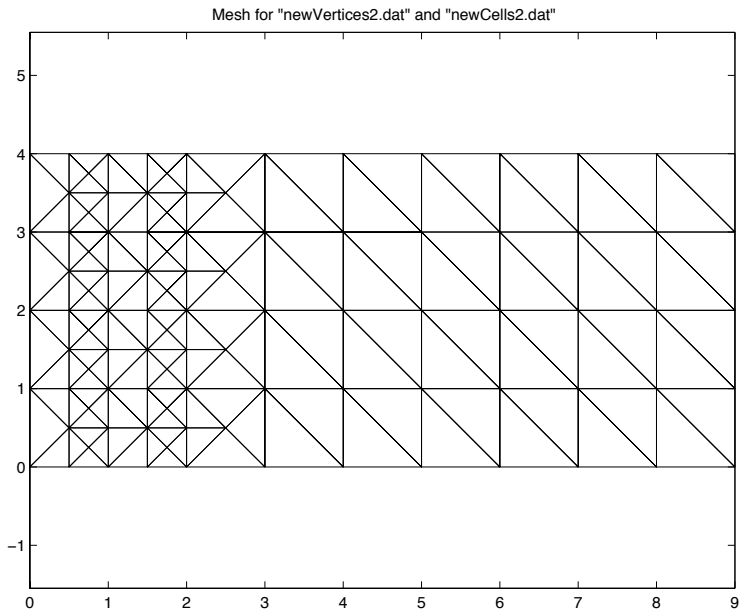

Figure 17: Output 1 of TC4

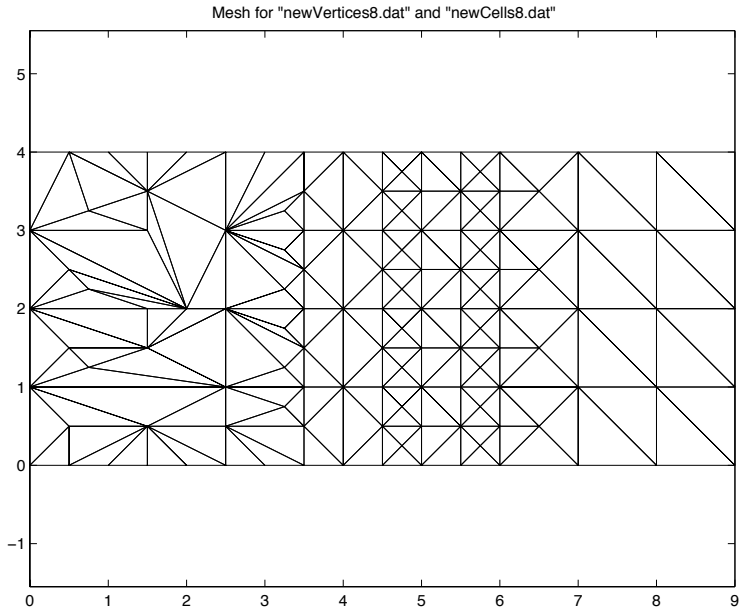

Figure 18: Output 2 of TC4

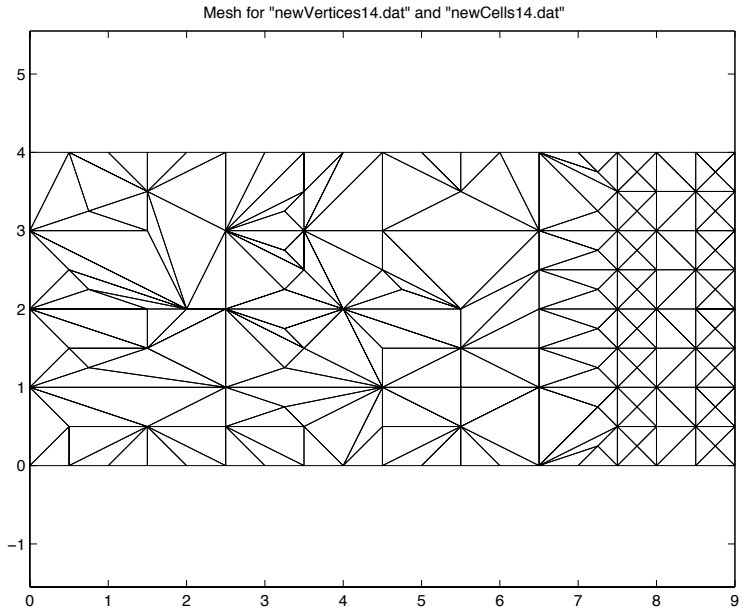

Figure 19: Output 3 of TC4

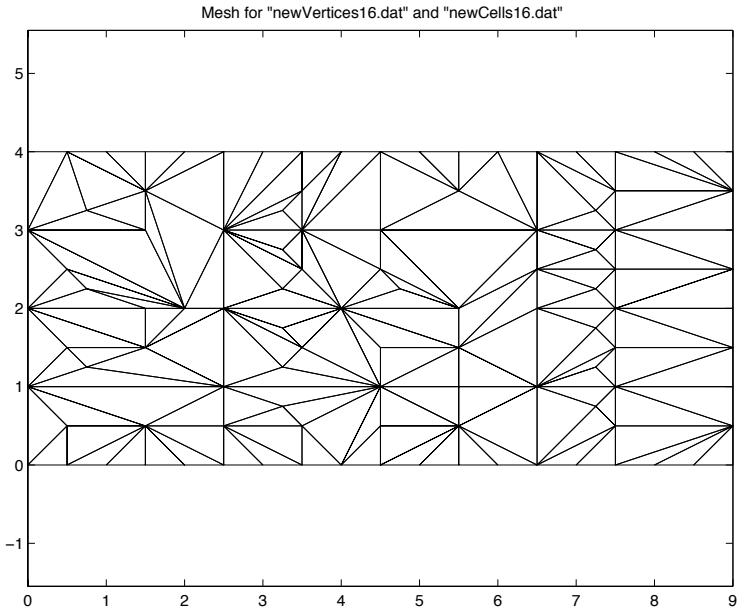

Figure 20: Output 4 of TC4

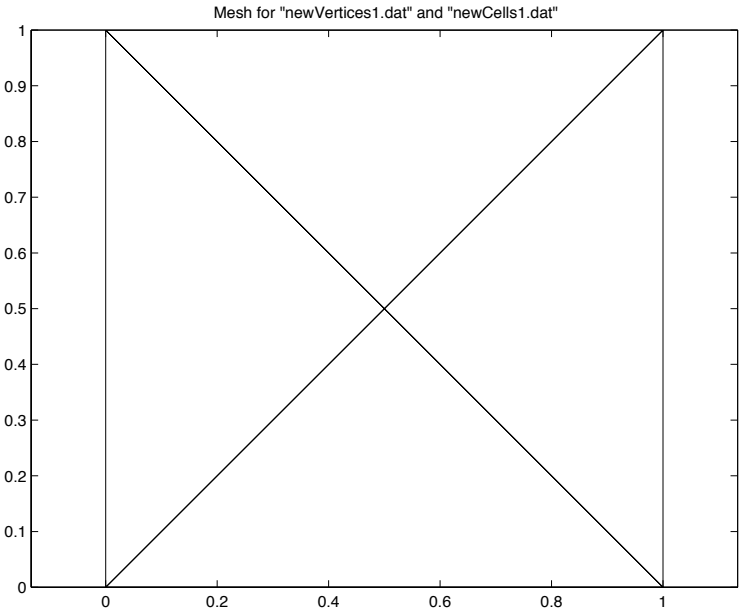

Figure 21: Output 1 of TC5

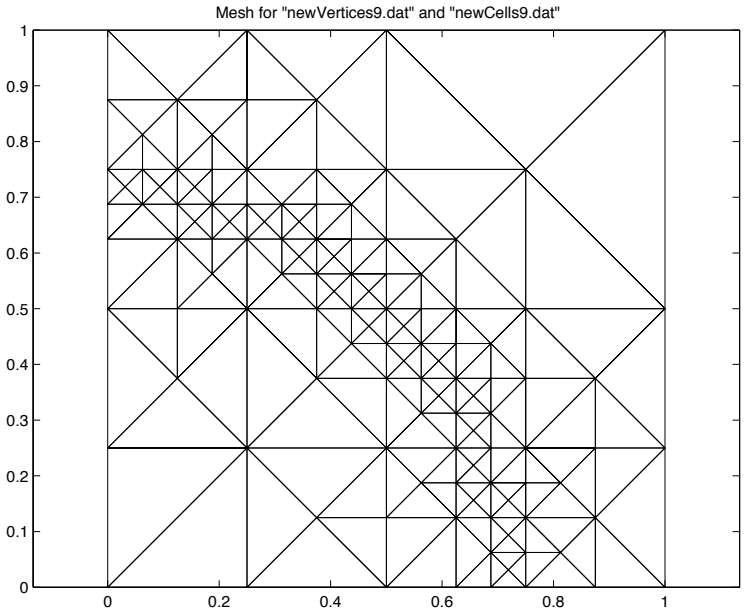

Figure 22: Output 2 of TC5

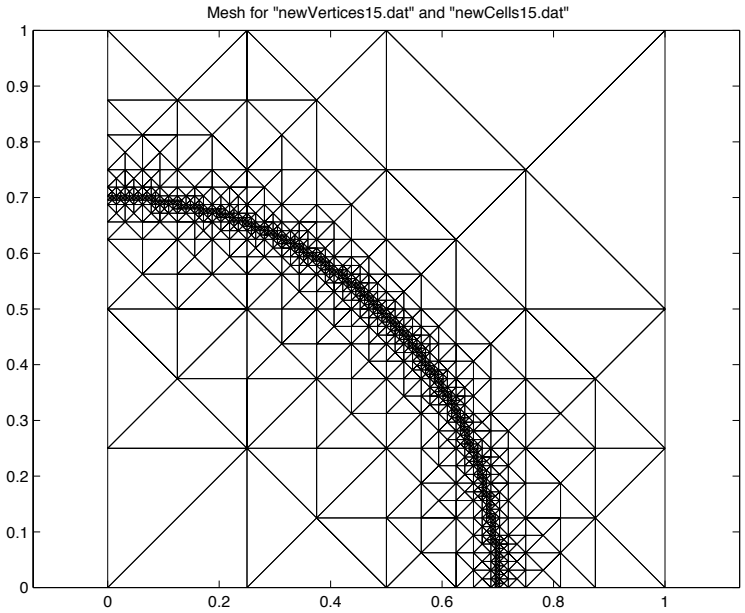

Figure 23: Output 3 of TC5