Mesh Untangling

(Raw material for the report)

August 9, 2002

Pavel Váchal
(vachal@galileo.fjfi.cvut.cz)

Mikhail Shashkov

Rao Garimella

Contents

Chapter 1

1D

1.1 Description of Untangling Methods

1.1.1 Feasible Set (FS)

- Runs in loop through all mesh vertices left to right.
- If the feasible set of x_i is not empty and one of the neighbouring cells is invalid, places x_i into the middle of its feasible set.
- Stops if there are no changes during one cycle.
- The cell (x_i, x_{i+1}) is invalid, if $x_{i+1} - x_i < d$, where d is some fraction (e.g. 1/100) of average cell volume.

1.1.2 Global Optimization (OPT)

1. Finds the negative gradient of objective function

$$f_\beta = \sum_{i=1}^{3} \left(|x_i - x_{i-1} - \beta| - (x_i - x_{i-1} - \beta) \right)^2 \tag{1.1}$$

   over the current mesh.

2. Along this direction, finds minimum of the objective function. Uses interval refinement method.
3. In minimum point repeats from step 1.

- Stops if
  - relative change during the last iteration (L2-norm of difference before and after divided by L2-norm before) is smaller than some constant , or
  - the gradient norm is smaller than some constant.

1.1.3 Three - stage algorithm (FS ∘ OPT ∘ FS)

1. First applies the feasible set approach. This fixes all vertices with non-empty feasible set, without disturbing rest of the mesh.
2. If there are still bad cells, uses the global optimization.
3. Finally, some of possibly created zero volume elements can be fixed by applying feasible set method once more.

1.2 Tests

Figs. 1.2 to 1.9 are to be understood as follows: Upper row shows the progress of feasible set approach, middle row progress of global optimization and the lower row is for the three-stage algorithm. In the left column, each color line shows the movement of one vertex. The y-axis shows number of current step or iteration. If the final (i.e. upper) ordering of markers corresponds with the legend, the mesh is untangled. In the right column of plots, each line represents the whole mesh at some moment of computation. For feasible set (top), snapshot is made each time a vertex is moved. For optimization (middle), snapshot after selected iterations are shown. For three-stage algorithm, the first (upper) snapshot is the initial tangled mesh, followed by one snapshot after each stage. (Thus, there are always four snapshots.)

1.2.1 1D Mesh of 4 Vertices

	x_1	x_2	x_3	x_4
test 4-1	0.0	-2.0	3.0	1.0
test 4-2	0.0	-2.0	0.5	1.0
test 4-3	0.0	-1.5	-2.0	1.0
test 4-4	0.0	0.0	-2.0	1.0
test 4-5	0.0	3.0	-2.0	1.0

Table 1.1: *Initial conditions for 1D tests with 4 mesh vertices.*

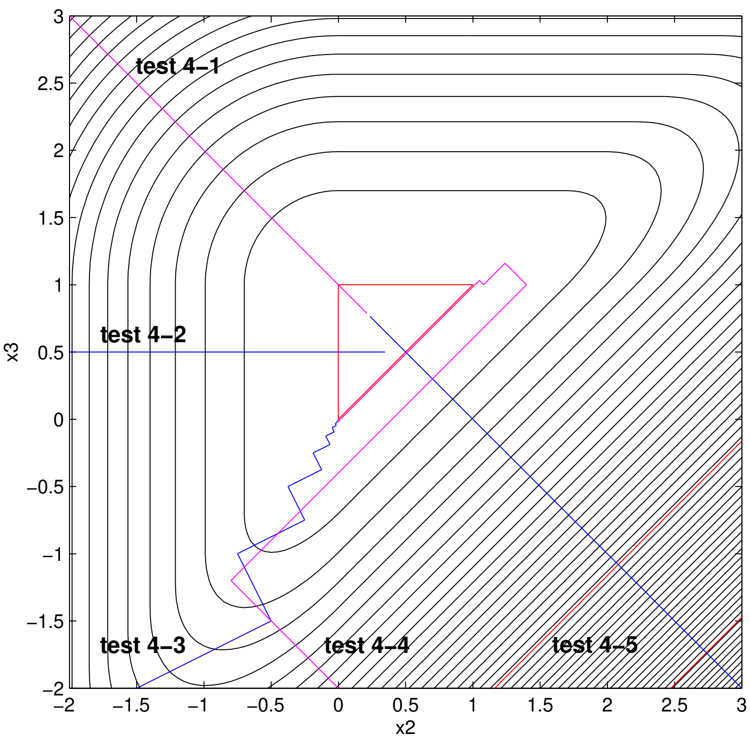

Figure 1.1: *Dependency of the objective function on position of vertices x_2 and x_3. Minimum $f_{obj} = 0$ inside the red triangle. Blue and magenta lines display results of optimization for tests 4-1 to 4-5.*

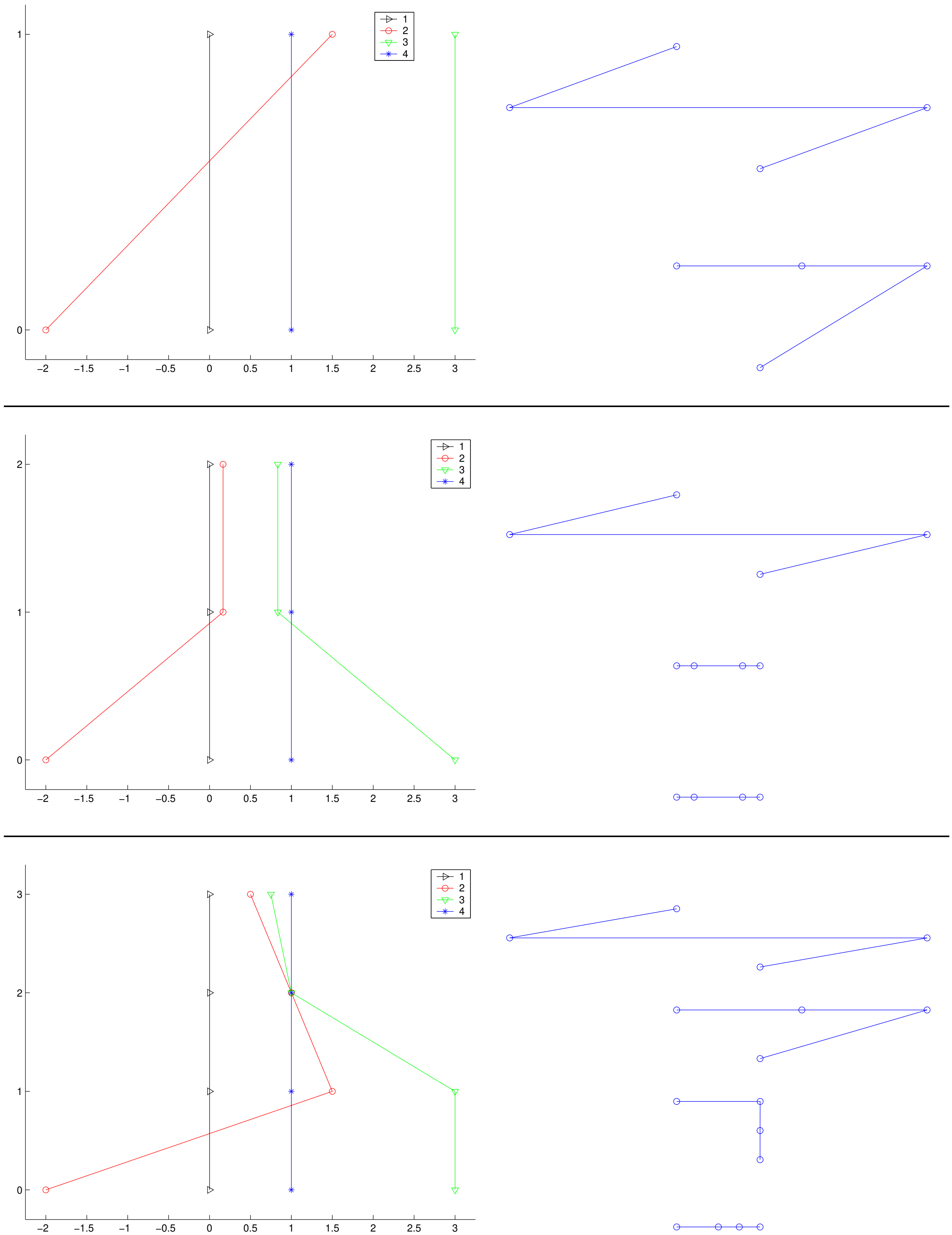

Figure 1.2: *Test 4-1. Top: FS. Middle: OPT. Bottom: $FS \circ OPT \circ FS$. Legend shows indices of x.*

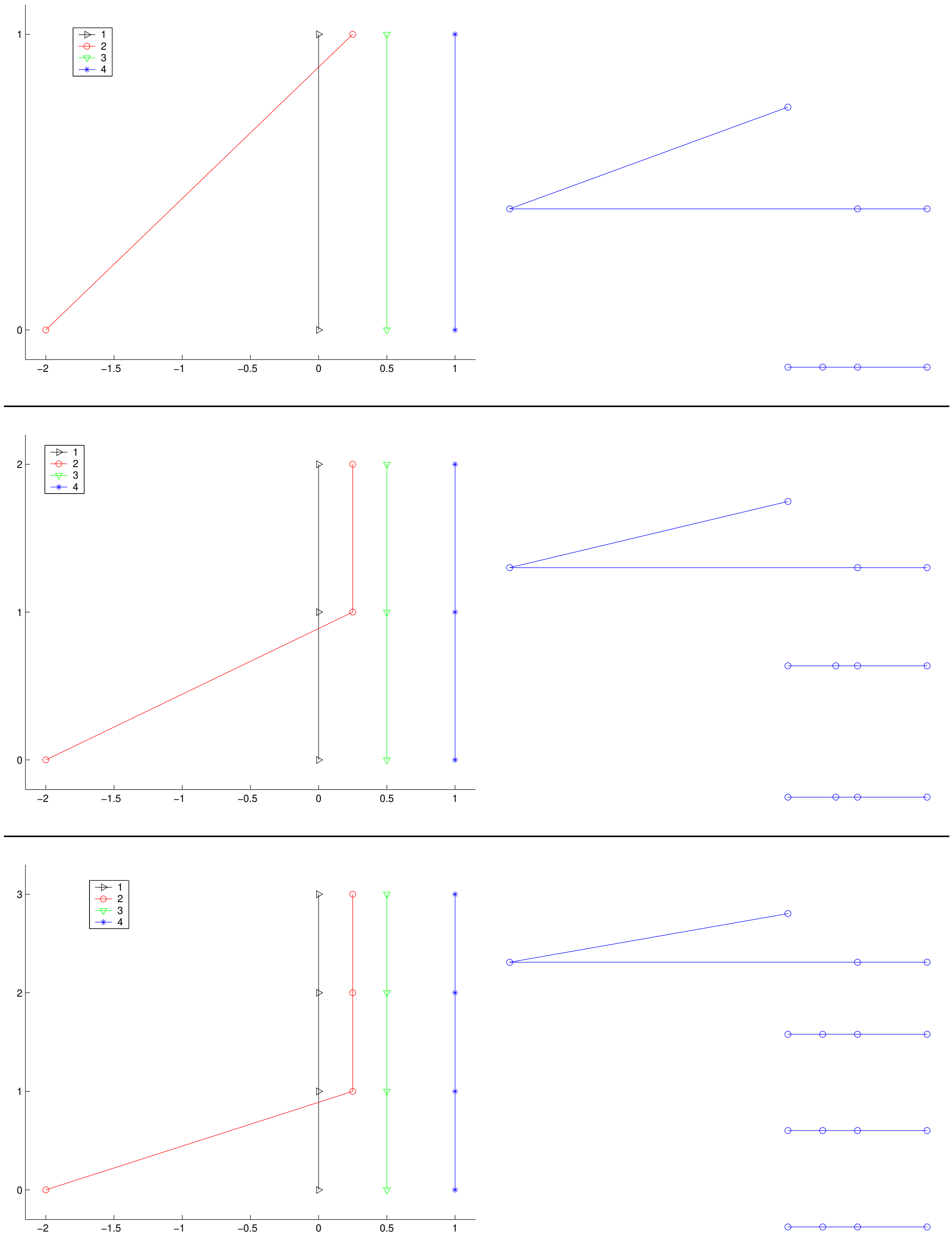

Figure 1.3: *Test 4-2. Top: FS. Middle: OPT. Bottom: $FS \circ OPT \circ FS$. Legend shows indices of x.*

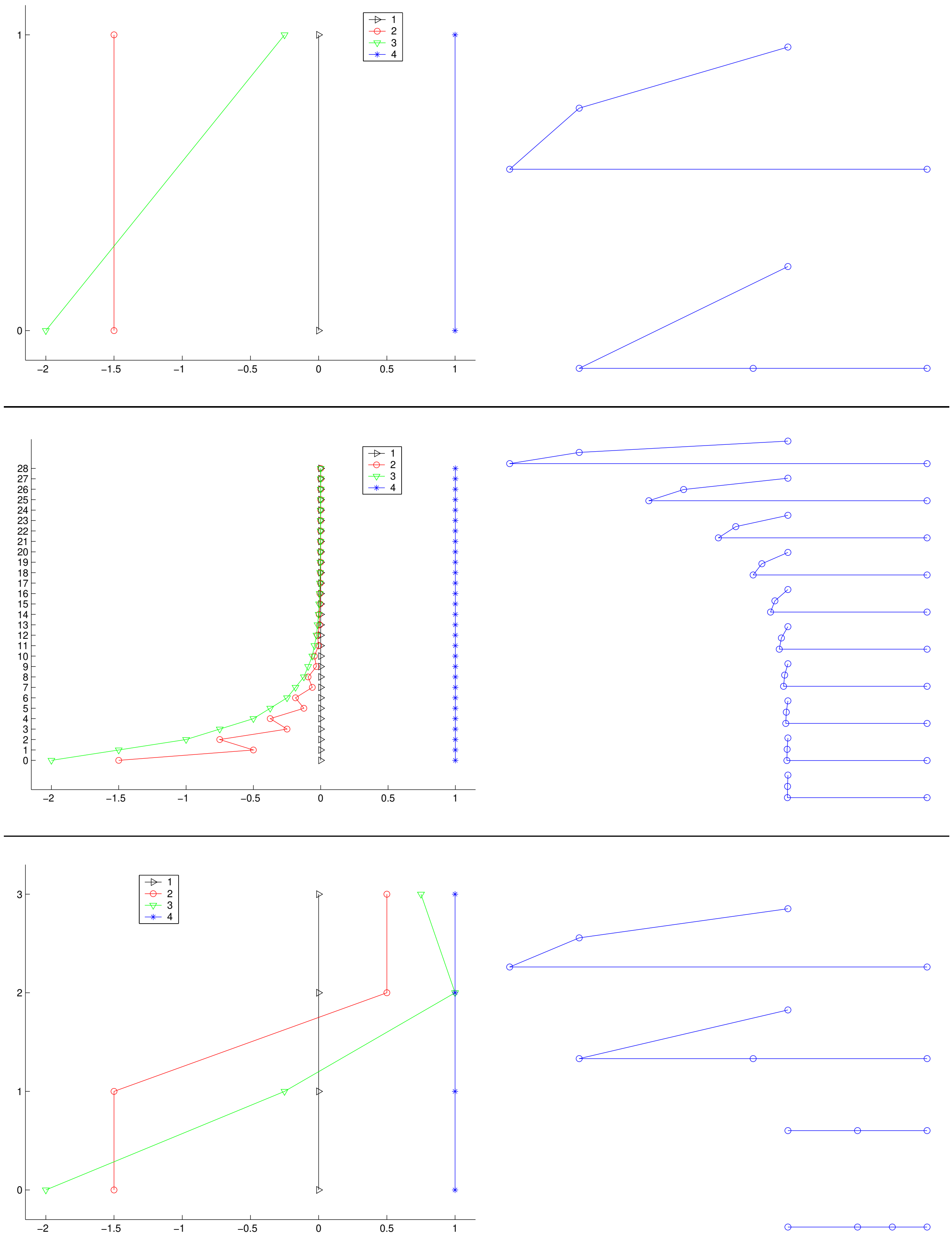

Figure 1.4: *Test 4-3. Top:* FS*. Middle:* OPT*. Bottom:* $FS \circ OPT \circ FS$*. Legend shows indices of* x*.*

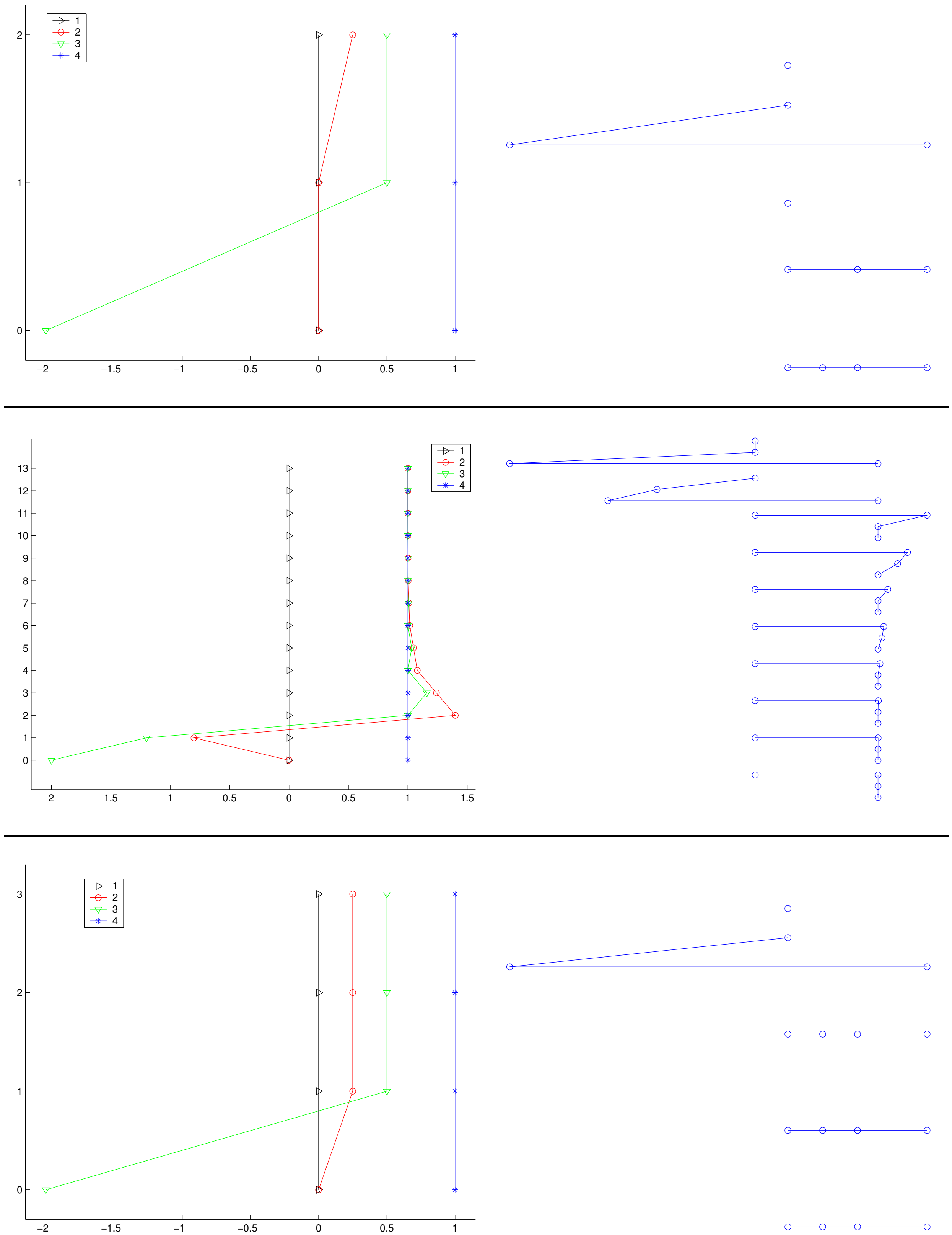

Figure 1.5: *Test 4-4. Top: FS. Middle: OPT. Bottom: $FS \circ OPT \circ FS$. Legend shows indices of x.*

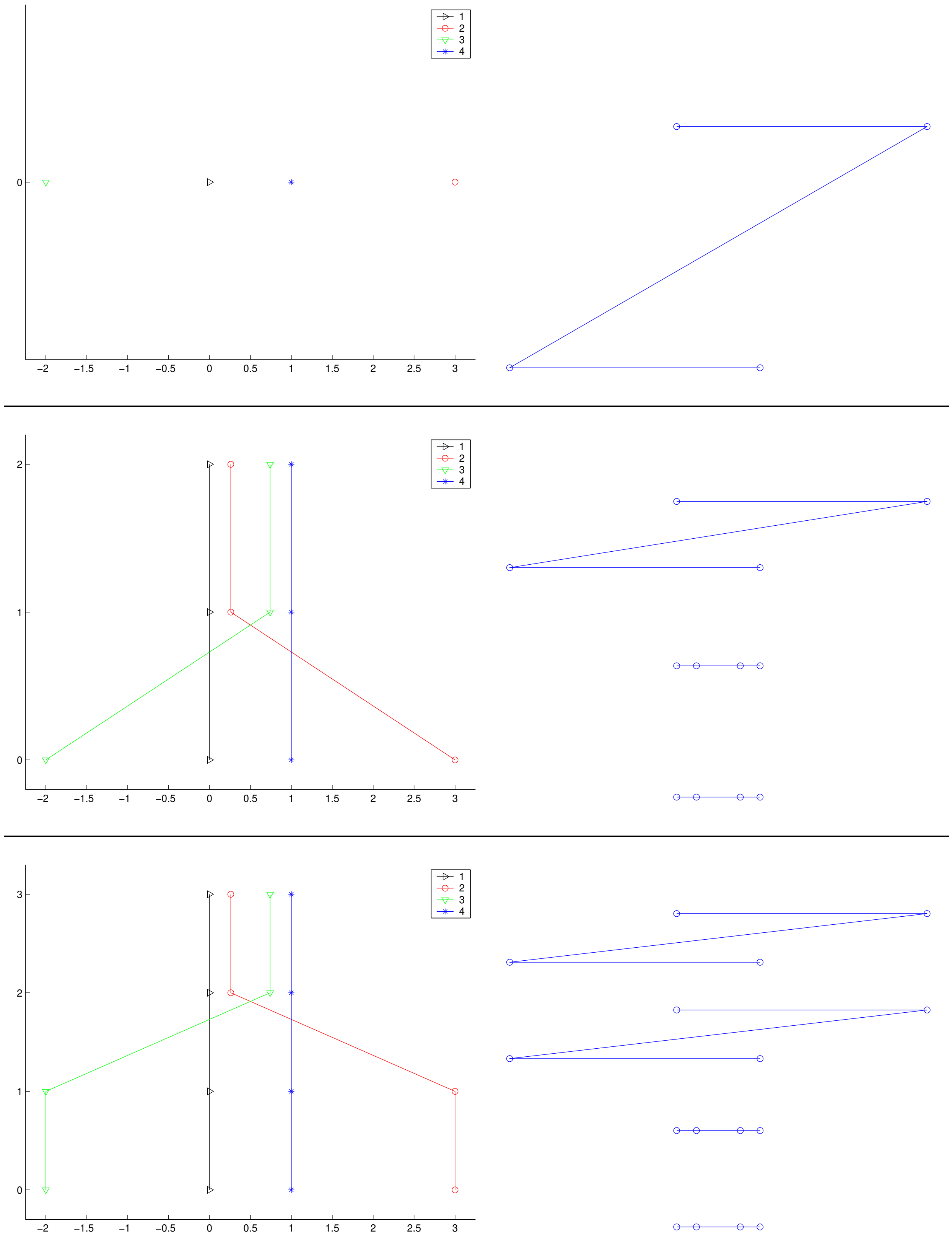

Figure 1.6: *Test 4-5. Top:* FS. *Middle:* OPT. *Bottom:* $FS \circ OPT \circ FS$. *Legend shows indices of* x.

1.2.2 1D Mesh of 10 Vertices

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
test 10-1	1.0	2.0	3.0	4.0	5.0	35.0	7.0	8.0	9.0	10.0
test 10-2	1.0	2.0	4.0	3.0	5.0	7.0	9.0	6.0	8.0	10.0
test 10-3	1.0	9.0	8.0	7.0	6.0	5.0	4.0	3.0	2.0	10.0

Table 1.2: *Initial conditions for 1D tests with 10 mesh vertices.*

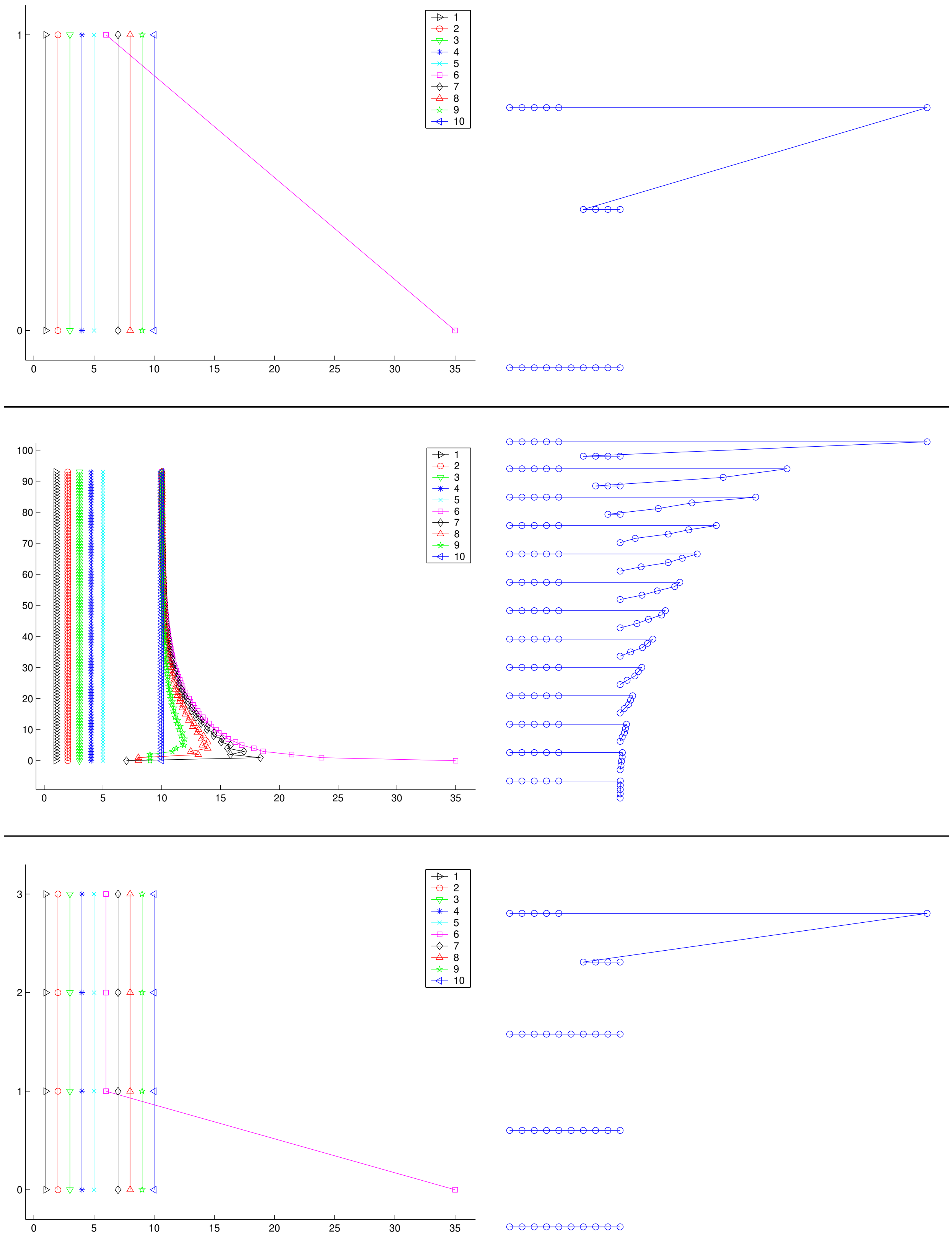

Figure 1.7: *Test 10-1. Top: FS. Middle: OPT. Bottom: $FS \circ OPT \circ FS$. Legend shows indices of x.*

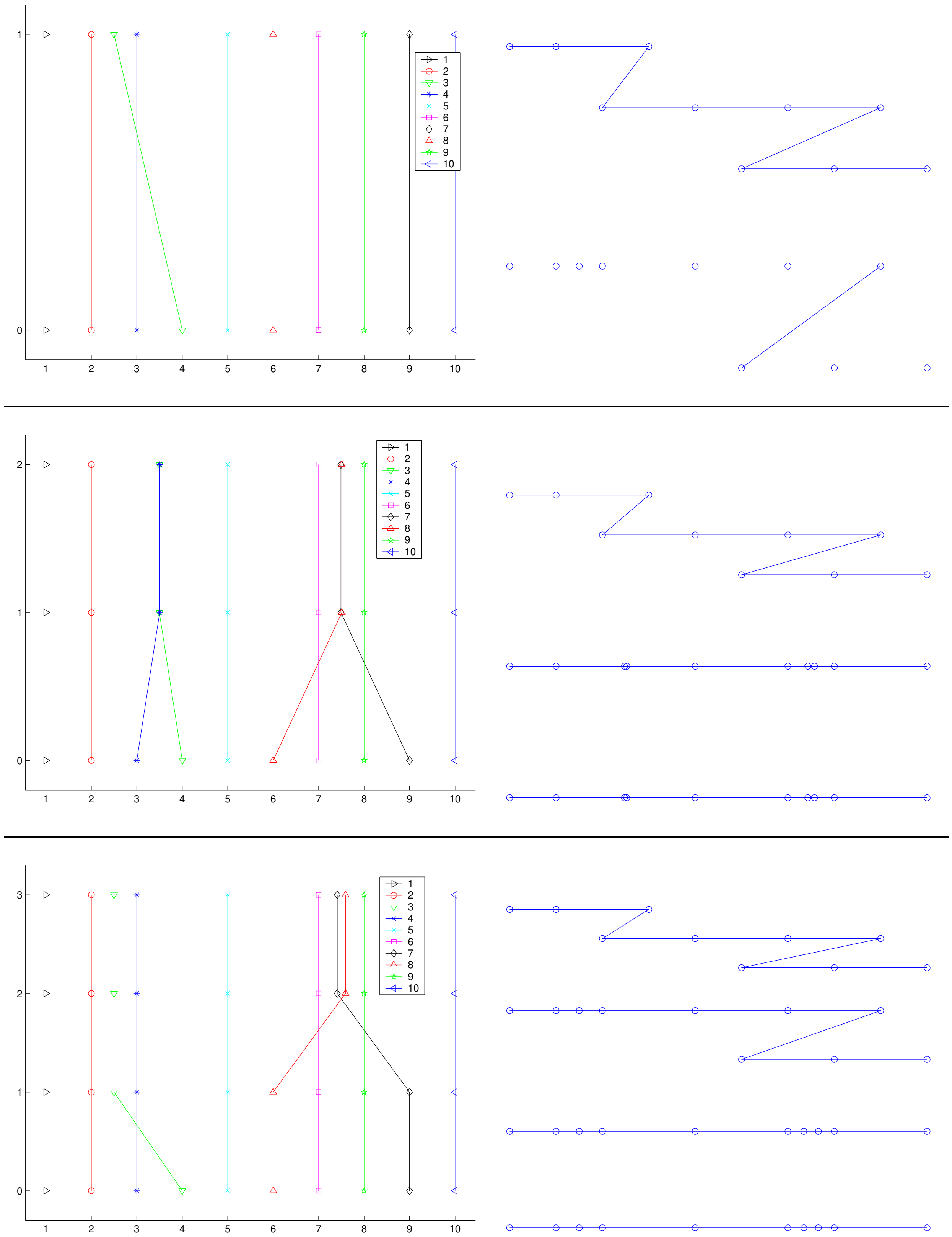

Figure 1.8: *Test 10-2. Top: FS. Middle: OPT. Bottom: $FS \circ OPT \circ FS$. Legend shows indices of x.*

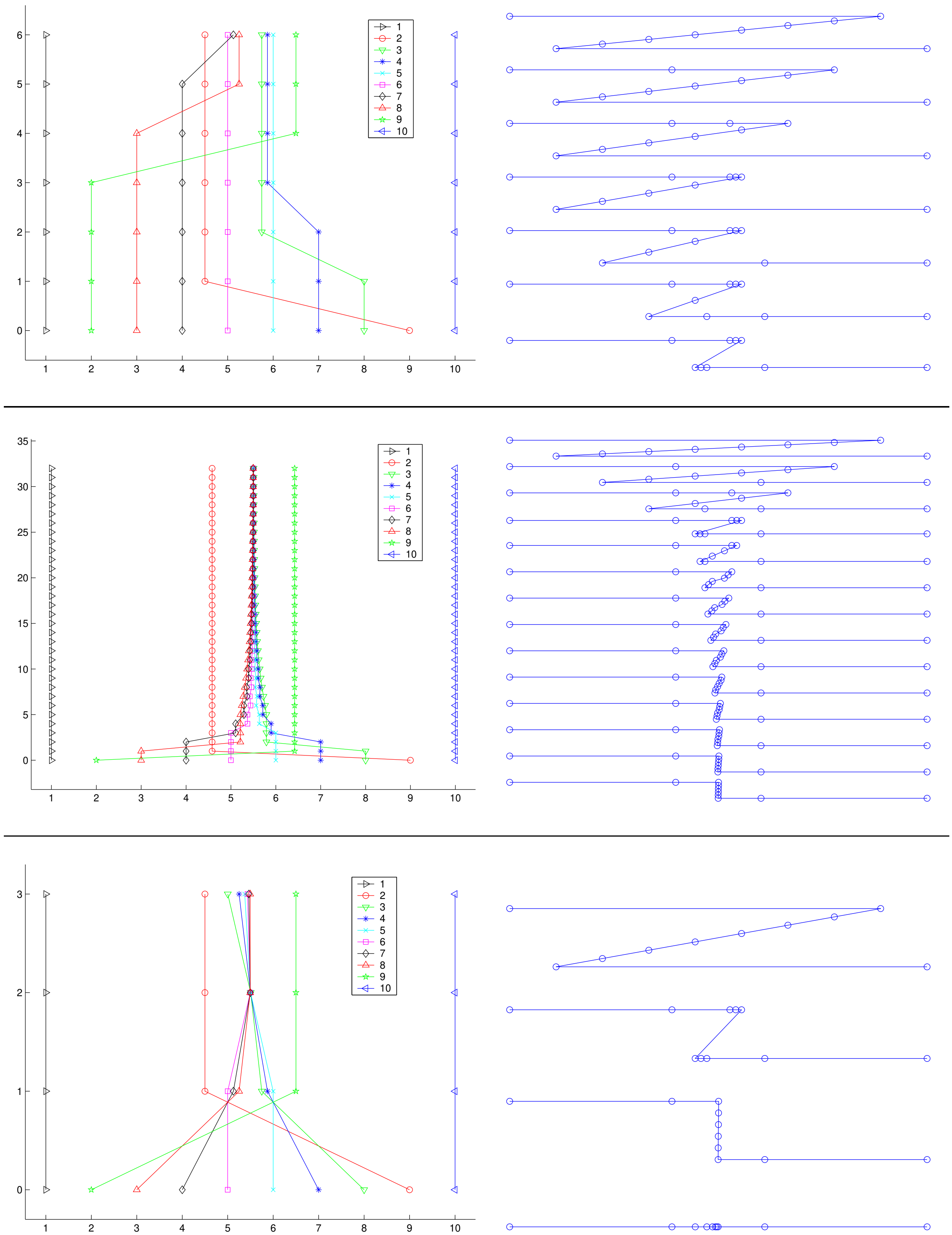

Figure 1.9: *Test 10-3. Top:* FS*. Middle:* OPT*. Bottom:* $FS \circ OPT \circ FS$*. Legend shows indices of* x*.*

Chapter 2

2D

2.1 Description of Untangling Methods

2.1.1 Feasible Set via Intersections (FS)

- Runs in loop through all patches.
- If the patch is bad and feasible set of its central vertex is not empty, places this vertex to the center of its feasible set.
- Stops if there are no changes during one cycle, or if the vertices start to oscillate.

- The patch is bad, if
  - at least one of connected cells is invalid, or
  - central vertex is too close to some of the surrounding vertices (this minimum required distance is related to size of the domain)
- Feasible set of a vertex with respect to its patch can be found by computing intersections of all linear constraints, imposed on it by the surrounding vertices. Example for structured quadrilateral and unstructured triangular patch can be seen in Figs. 2.1 and 2.2.

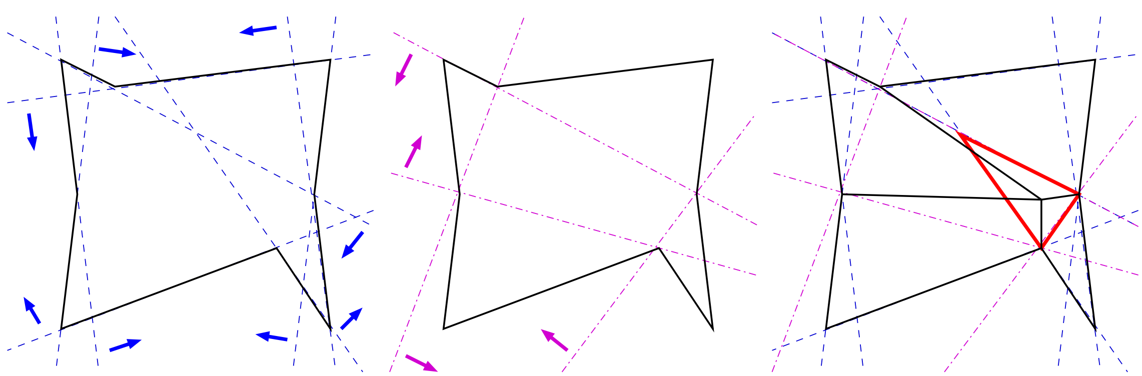

Figure 2.1: *Feasible set (red) as an intersection of linear constraints (blue and magenta) for structured quadrilateral mesh. Untangled patch shown in black in the last figure.*

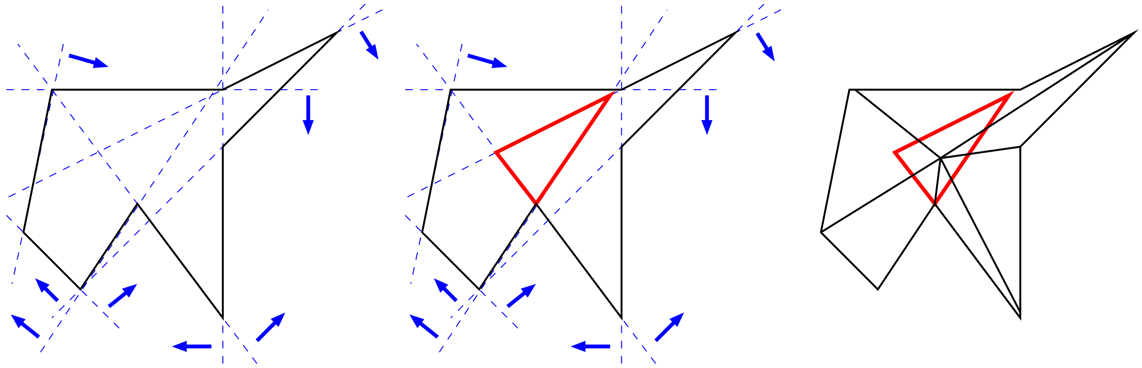

Figure 2.2: *Feasible set (red) as an intersection of linear constraints (blue) for unstructured triangular mesh. Untangled patch shown in black in the last figure.*

2.1.2 Feasible Set via Simplex Method (SIM)

Analytical computation of intersection polygon of all linear constraints is very time consuming (and in fact, we don't have it implemented in 3D). To make the procedure faster, we can use simplex method in such a way, that is guaranteed to find a subset of this polygon. In particular, we optimize four linear functions: $f = x$, $f = -x$, $f = y$ and $f = -y$. If the feasible set is not empty, this finds its subset. Particularly, if the area of feasible set is nonzero, a triangular or quadrilateral subset will be found (see Fig. 2.3). If the feasible set is a line or a single point, simplex method approach finds at least one point.

Since it is not necessary to find exactly the centroid of feasible set (in fact it would be enough to find an arbitrary point inside), we can place the vertex into the center of that subset, which we have found with the simplex method.

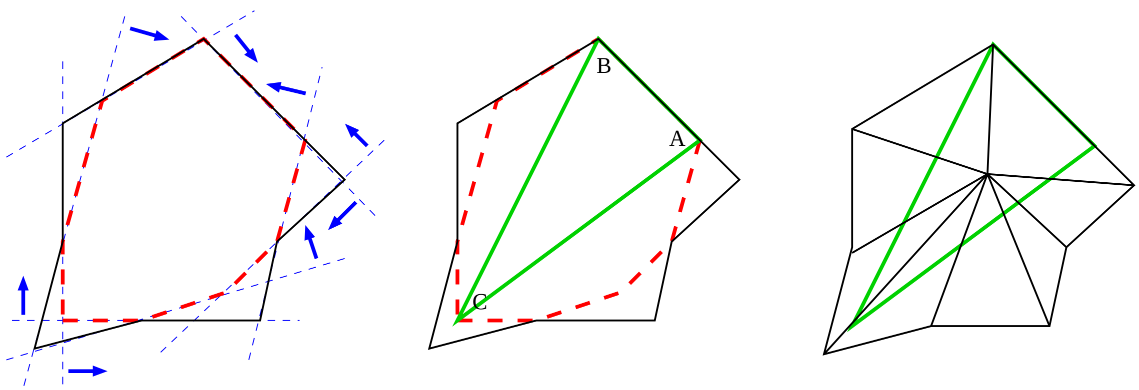

Figure 2.3: *Using the simplex method to untangle an unstructured triangular mesh. By applying simplex method on four different functions, we get some subset (green) of the feasible set (red dashed). In particular, vertex A is obtained by maximizing $f = x$, B by $f = y$ and C with $f = -x$ as well as $f = -y$. Untangled patch is shown in black at far right.*

2.1.3 Local Optimization (LOPT)

1. For each vertex, finds the negative value of gradient of objective function

$$f_\beta = \sum_{i=1}^{N_c} \left(|A_i - \beta| - (A_i - \beta)\right)^2 \tag{2.1}$$

   On unstructured traingular meshes, N_c is the number of cells connected to this vertex and A_i are areas of these cells. (If the cell is inverted, its area is negative.) For structured quadrilateral meshes, N_c is equal to 12 and A_i are areas of the first twelve triangles from Fig. 2.4. Parameter β will be discussed later.

2. Along direction given by this negative gradient (called steepest descent direction), finds minimum of the objective function. Uses interval refinement method. Moves the vertex to the position, which corresponds to this minimum.

3. If the patch is fixed, if gradient norm is too small or if the vertex did not move enough during the last iteration, goes to the next patch. Else repeats from step 1.

4. This repeats until all patches are fixed or until the vertices stop to move. For details on break conditions, see the manual for particular codes.

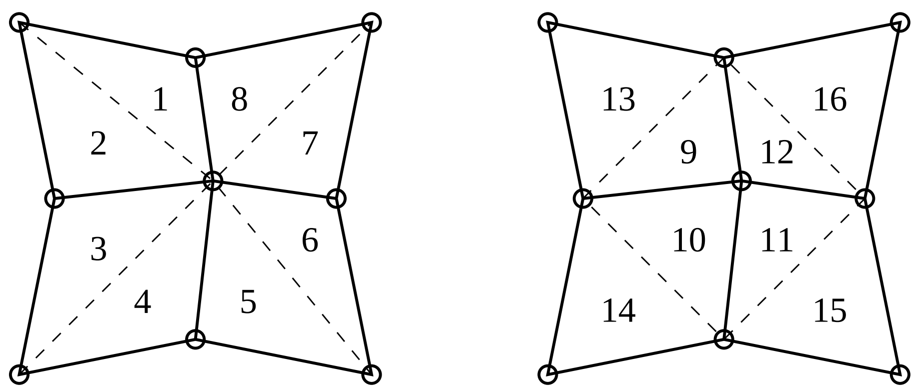

Figure 2.4: *Splitting four quadrilateral cells of an structured mesh into triangles.*

2.1.4 Global Optimization (GOPT)

1. Finds the negative value of gradient of global objective function. This function can be considered as a sum of objective functions over all patches. Because of efficiency, we sum over cells rather than over patches, i.e. we take each triangle only once.

2. Along direction given by this negative gradient (called steepest descent direction), finds minimum of the objective function. Uses interval refinement method. Moves the vertices to positions, which correspond to this minimum.

3. Repeats from step 1 until one of the following situations occurs: (1) all patches are fixed, (2) gradient norm is too small or (3) the vertices did not move enough during the last iteration.

   For details on break conditions, see the manual for particular codes. Note, that some of these three expensive tests can be omitted, if (I) we know, that the solution exists, (II) we know, that the objective function is convex, (III) minimum of objective function is zero and is reached if and only if the mesh is untangled and (IV) definition of a "good patch" allows zero volume elements.

2.1.5 Three - stage algorithms

1. First applies one of the feasible set approaches (FS or SIM). This fixes all vertices with non-empty feasible set, without disturbing rest of the mesh.
2. If there are still bad cells, uses the global optimization.
3. Finally, some of possibly created zero volume elements can be fixed by applying one of feasible set methods once more.

Although local optimization is also a local approach, it would not be of much sense to use it in the 3-stage algorithm:

- In step 3, we only want to fix zero volume cells. This can not be done with local optimization. More precisely, we would have to use non-zero parameter β, but then why not to use it already in step 2?.
- In step 2, the main goal is to fix all elements. Though it seems, that with quadratic objective function the patches will not oscillate (i.e. destroy another patch by fixing some vertex and vice-versa) even if treated separately, we did not prove this analytically. Thus we prefer global optimization, which takes into account all patches at once.
- If we would apply local optimization in step 1, step 2 would probably be needless. We still did not find an example, where local optimization would fail to make all elements non-negative. On the other hand, global optimization seems to produce fewer zero volume cells.

2.2 Note on Optimization Parameter β.

If properly chosen, parameter β from (2.1) can avoid zero element volumes. Geometrically, it shinks the region, where objective function takes its minimum, towards the center of the feasible set. In other words, a vertex optimized with proper parameter β will not stop too close to the margin of feasible set, so that the patch will be not only “barely valid”.

However, even if we relate this parameter to size of the cell volume, as it is done in the original paper [1], its desired magnitude is still problem dependent. If β is too small, its effect is not strong enough. If it is bigger than some critical value, the minimum of objective function is outside the feasible set, which means that optimization produces an invalid patch (resp. mesh). This is clearly to see in Fig. 2.5.

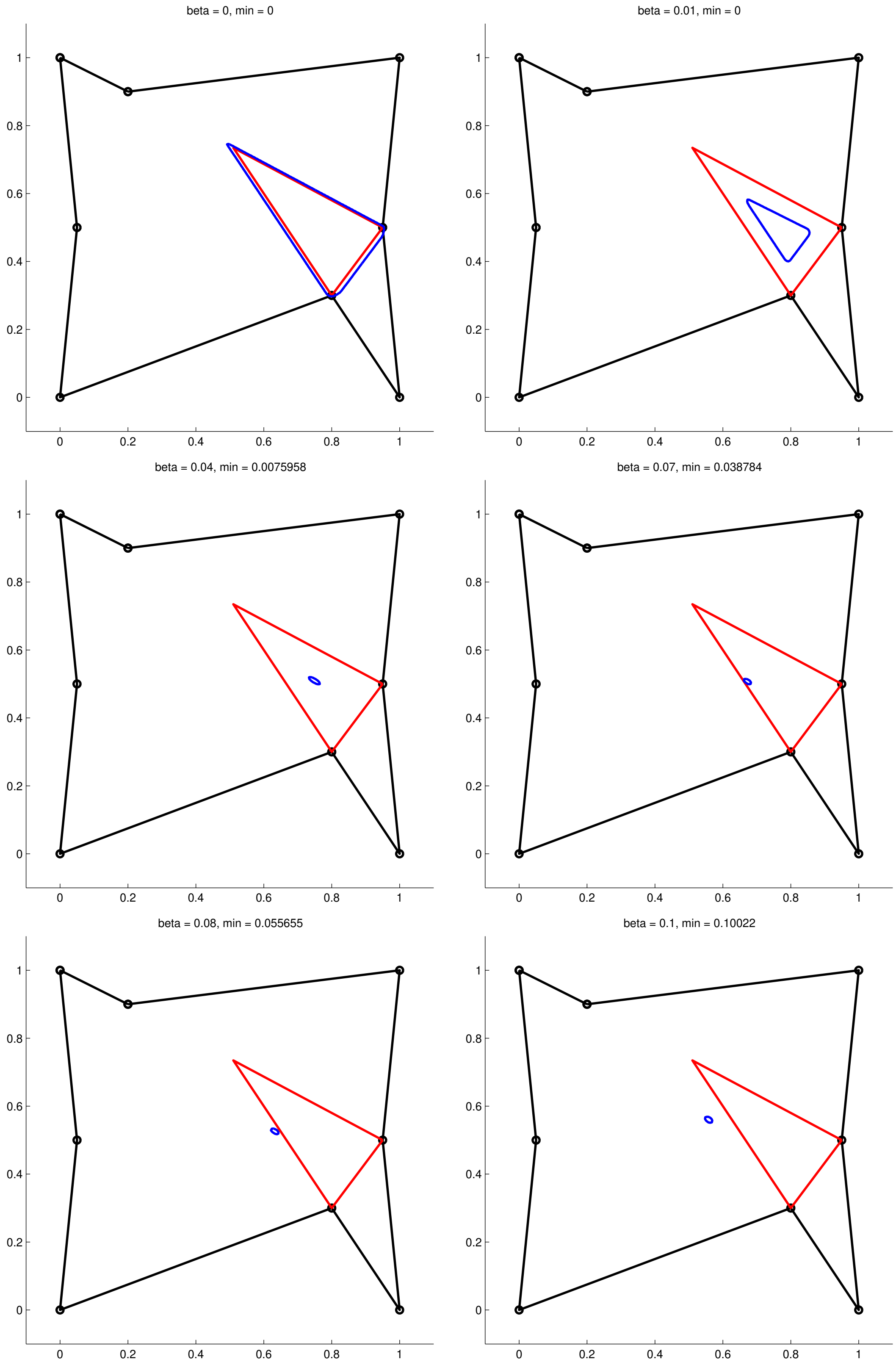

Figure 2.5: *Effect of optimization parameter β in one structured quadrilateral patch. With increasing β, minimum region of objective function (blue) shrinks inside the analytically computed feasible set (red) and then leaves it.*

2.3 Tests

Here we present results of six tests. Four of them are on a structured quadrilateral mesh, two on an unstructured triangular mesh.

2.3.1 Rotated Square Region of an Structured Mesh (2D-1 and 2D-2)

In both tests, we took a square-shaped region of an rectangular quadrilateral mesh, and rotated it with respect to the center. While untangling, we have moved only interior vertices.

In 2D-1, the rotation angle was $\pi/4$ (see Fig. 2.6). We can see in Fig. 2.7, that all methods succeded. Since local approaches (Feasible Set, Simplex Method and Local Optimization) are order dependent, they provided non-symmetric solutions. Note, that since Feasible Set and Simplex Method already untangled the mesh completely, the latter two steps were not needed. Thus the lower two plots are the same as the upper ones.

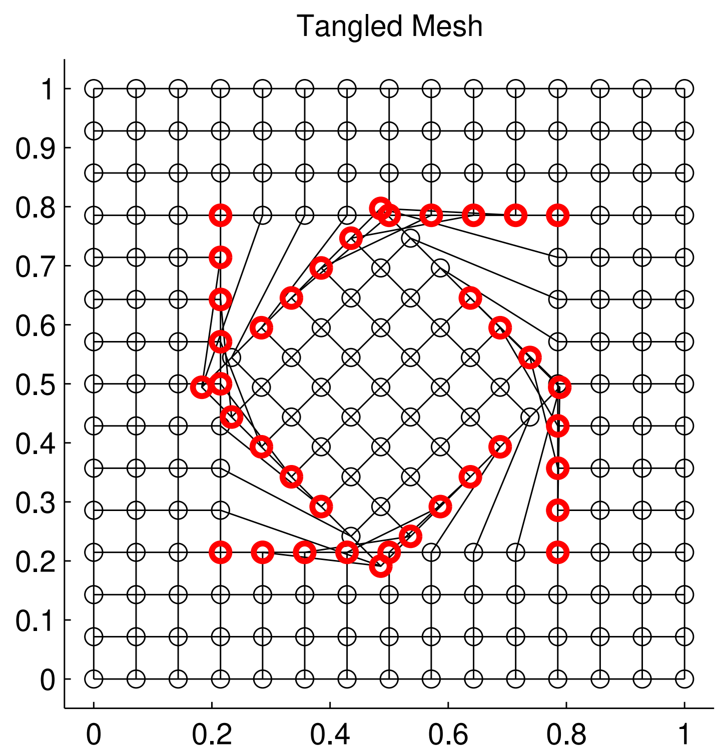

Figure 2.6: *Test 2D-1. Initial tangled mesh. Square rotated by $\pi/4$. Central vertices of bad patches are marked red.*

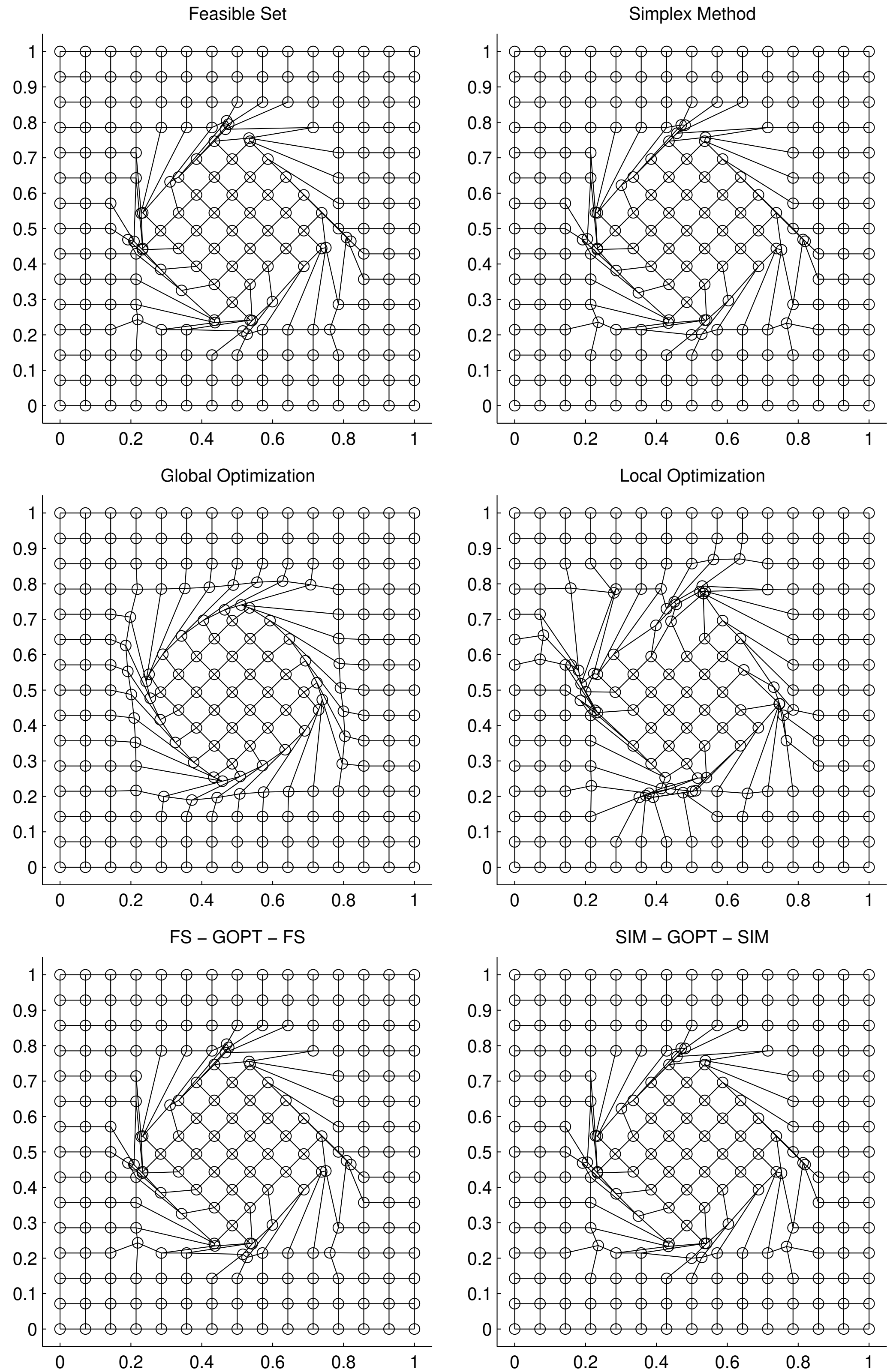

Figure 2.7: *Test 2D-1. Results of Feasible Set, Simplex Method, Local and Global Optimization and both Three-stage approaches.*

In 2d-2, we have rotated the rectangle by $3\pi/4$ (see Fig. 2.8). Results are shown in Fig. 2.9. Feasible sets of all bad vertices are empty and so neither Feasible Set method nor Simplex Method were able to untangle any single patch. Local Optimization made all cell volumes non-negative, but 29 of them stayed zero. Similarly, Global Optimization produced 18 zero elements. We see, that in a three-stage algorithm, these zero elements were fixed by applying Seasible Set once more (lower left plot).

Success of any feasible set - based method can depend on the choice, where do we place the vertex inside the feasible set. In our case, centroids of the feasible set differ from centroids of their subsets, found by Simplex Method. As shown in the last plot of Fig. 2.9, this caused the failure of Simplex Method. This can be seen in more detail in Fig. 2.9.

Note, that since the first application of feasible-based methods did not move anything, the status was the same for both Three-stage algorithms after first two stages.

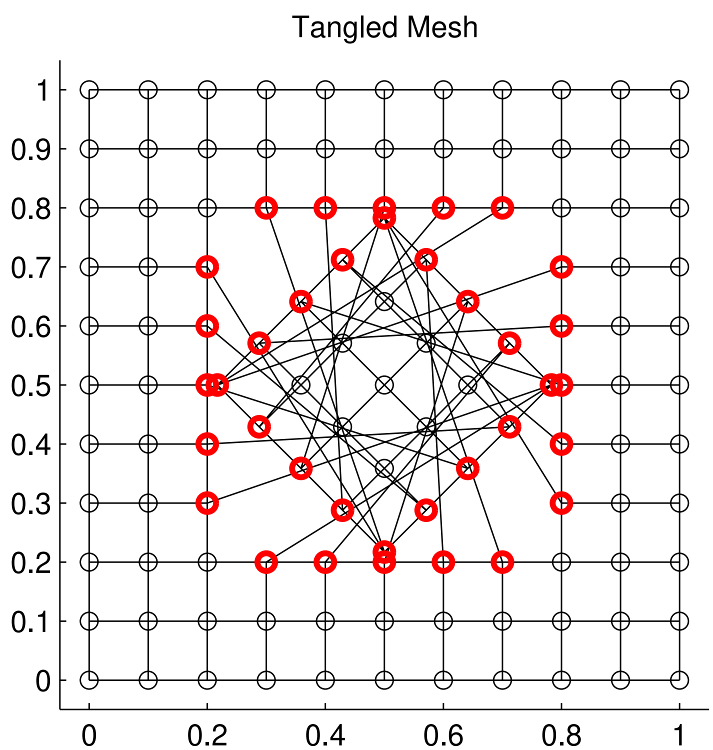

Figure 2.8: *Test 2D-2. Initial tangled mesh. Square rotated by $3\pi/4$. Central vertices of bad patches are marked red.*

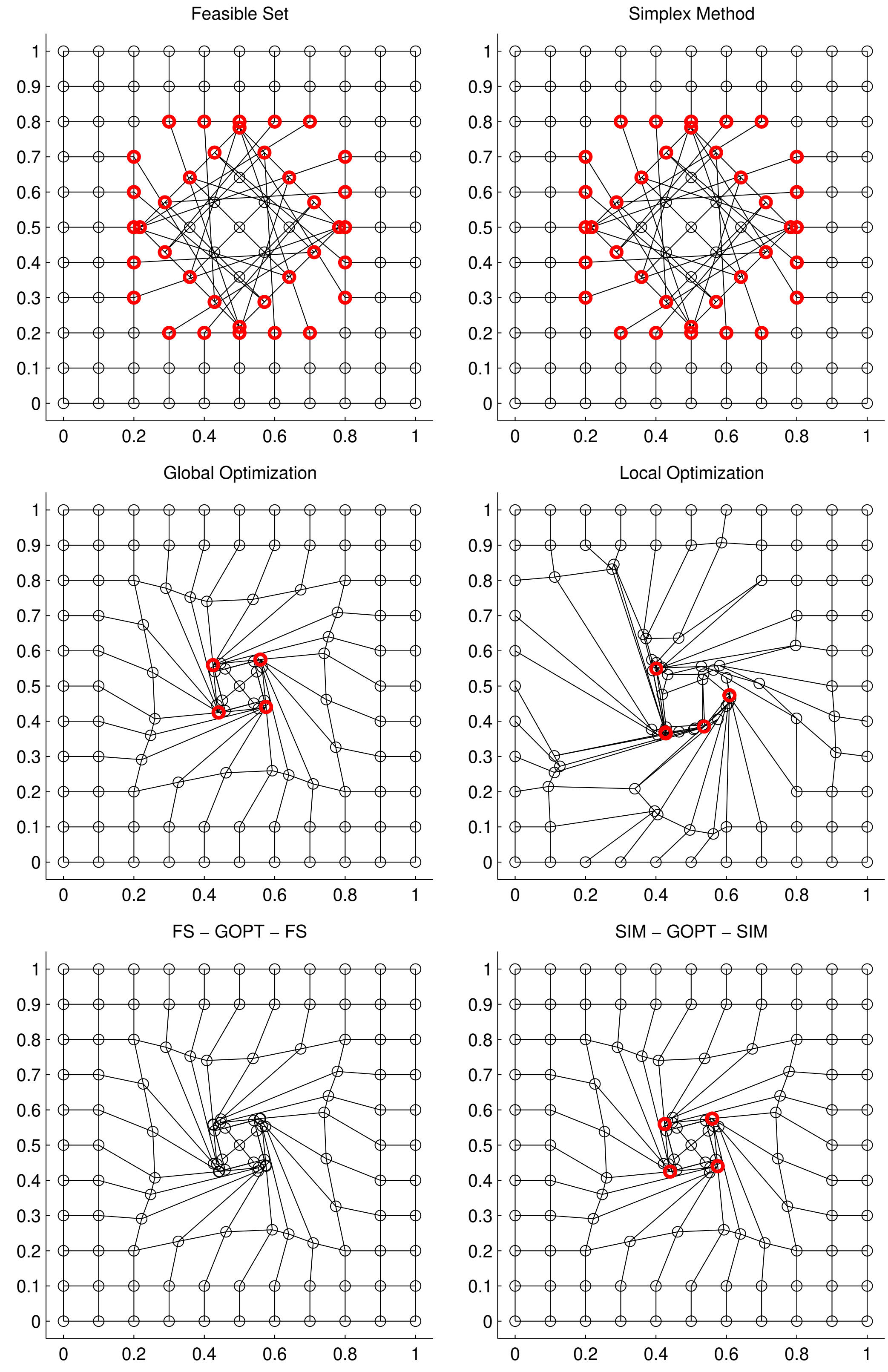

Figure 2.9: *Test 2D-2. Results of Feasible Set, Simplex Method, Local and Global Optimization and both Three-stage approaches. Central vertices of remaining bad patches are marked red.*

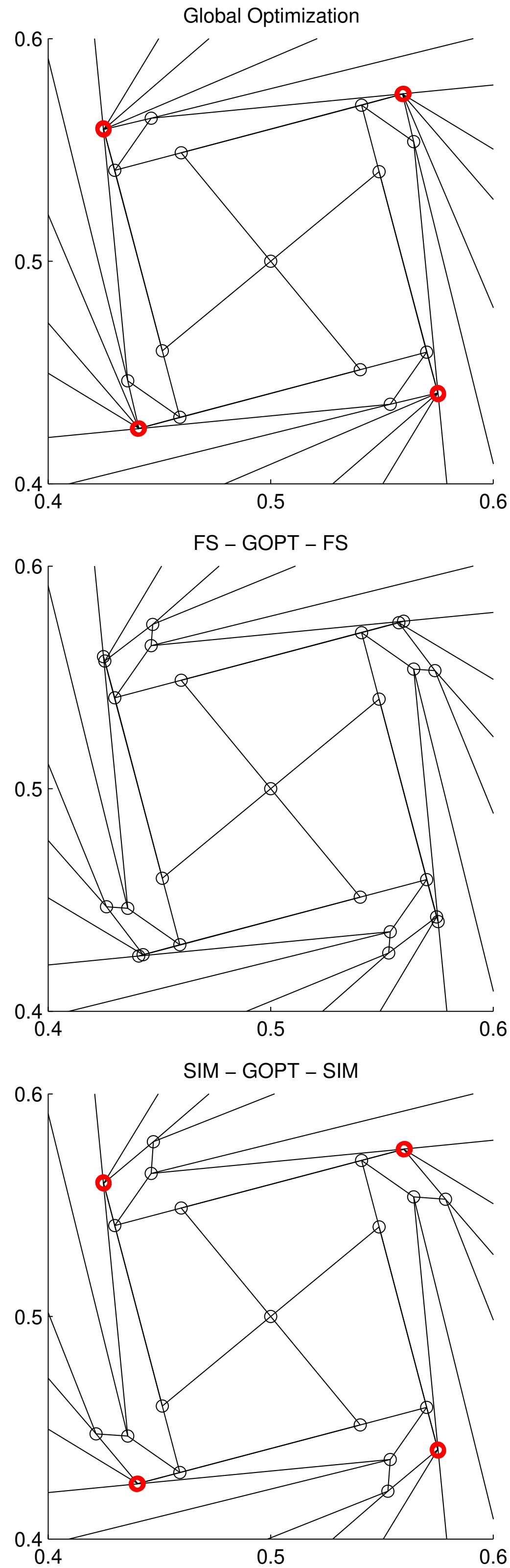

Figure 2.10: *Test 2D-2. Effect of the third stage in Three-stage algorithms. Central vertices of remaining bad patches are marked red.*

2.3.2 Tangled Horseshoe (2D-3)

In our next example, we took the horseshoe mesh from [2] and tangled it in the following way: In each round, each vertex is moved to the centroid of its four neighbors as computed after the last round. For test 2D-3, 30 tangling rounds have been made. Both the original and the tangled horseshoes are shown in Fig. 2.11.

While untangling, we allowed only interior vertices to move. Results can be seen in Fig. 2.12.

Both feasible set - based methods left 24 patches bad. Optimization approaches made all cell volumes non-negative, but left some of them zero. In particular, Local Optimization 10, Global Optimization 12.

In this case, zero elements were fixed by the three-stage algorithms. But typically, using horseshoe with other parameters, even these combined methods can leave some bad patches along the shorter arc inverted.

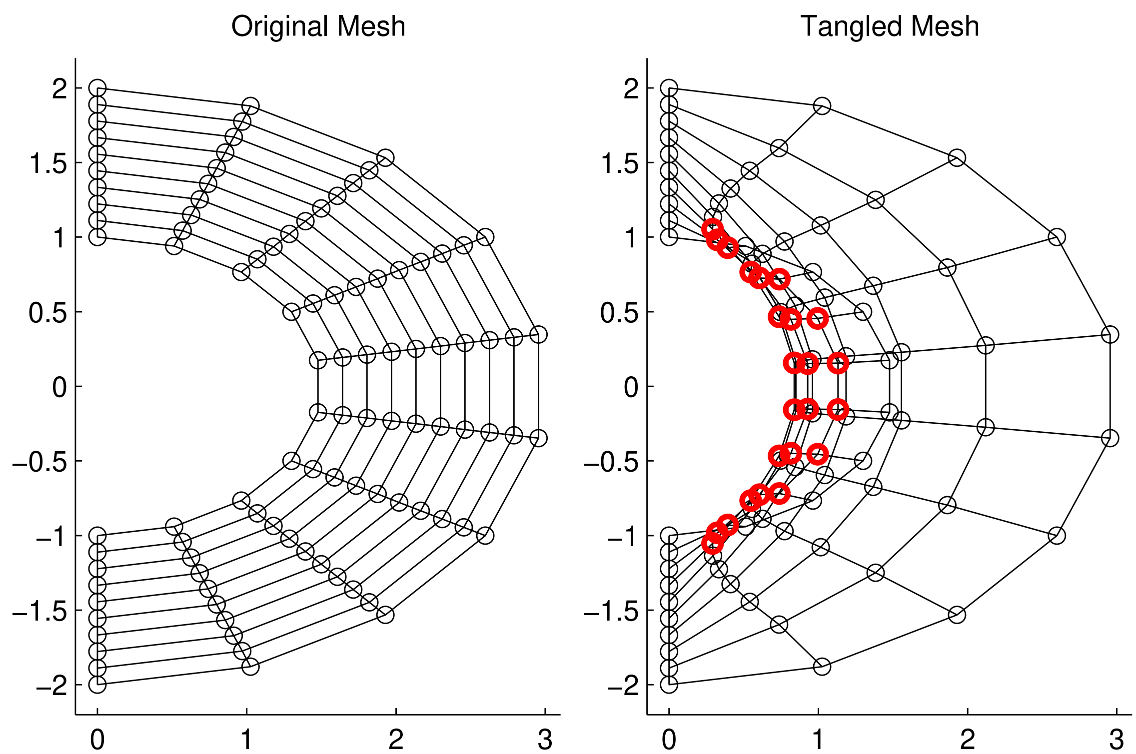

Figure 2.11: *Test 2D-3. Original and tangled horseshoe. Central vertices of bad patches are marked red.*

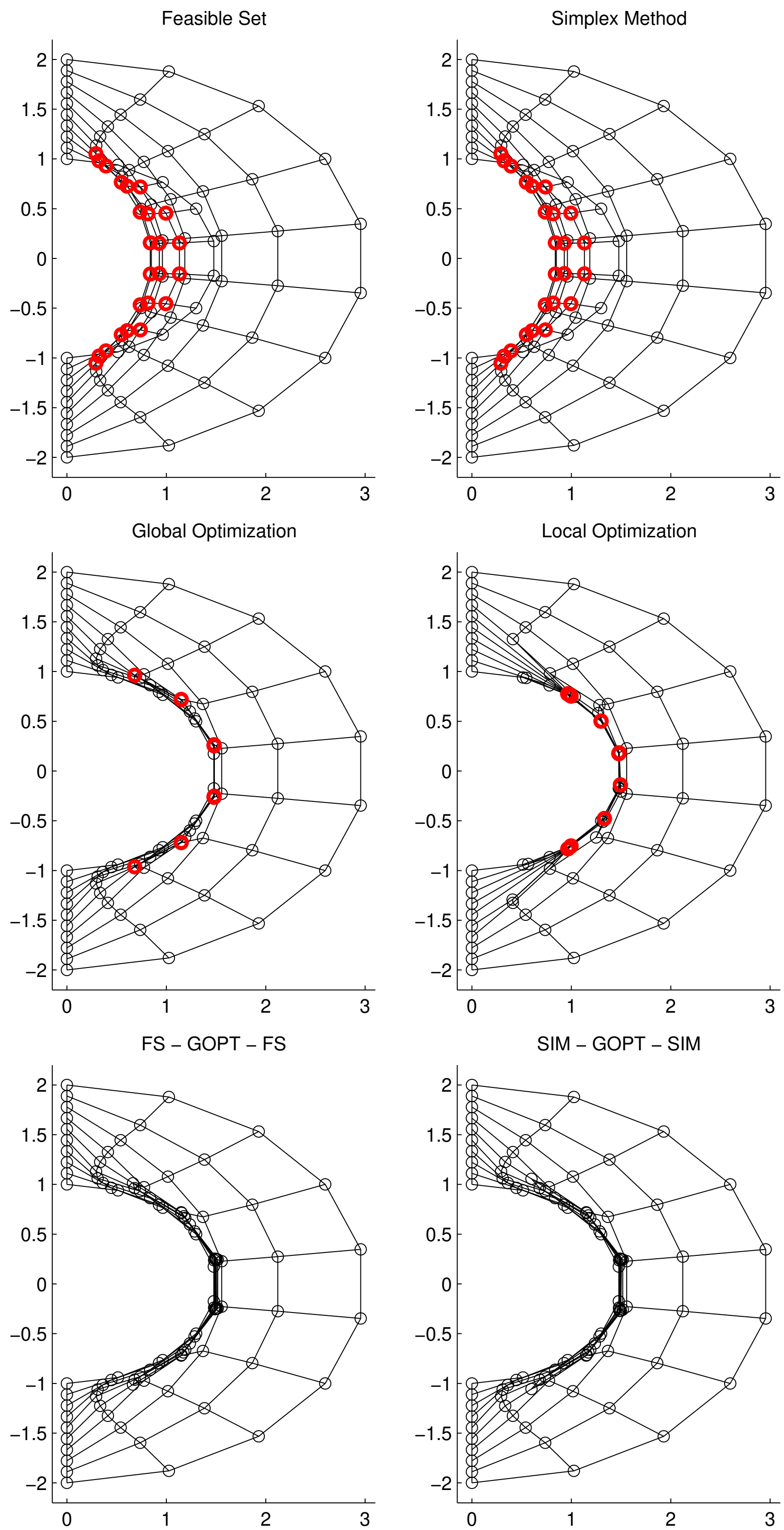

Figure 2.12: *Test 2D-3. Results of Feasible Set, Simplex Method, Local and Global Optimization and both Three-stage approaches. Central vertices of remaining bad patches are marked red.*

2.3.3 W - shaped mesh (2D-4)

We took another mesh from [2] and tangled it the same way as in previous example (see Fig. 2.13 top). Because of the sharp angle in the middle of the lower edge, this is even more difficult to untangle. Thus we will allow also the boundary vertices to move.

Both feasible set - based methods (see Fig. 2.13 bottom) have left five patches bad. Let's try the optimization (Fig. 2.13). Local Optimization of interior created 8 zero volume elements, Global Optimization of interior 5. If we combine Global Optimization of interior with Local Optimization of boundary (Let us denotttte this combination GL), mesh quality slightly improves, but there are still 5 zero volume elements.

The three-stage algorithms performed better, but if moving only interior vertices, both of them (i.e. that one with Feasible Set as well as that one with Simplex Method) still left one patch bad (Fig. 2.15 top). They have untangled the entire mesh, when we combined the Global Optimization of interior with the Local Optimization of boundary, i.e. when we have used GL instead of the second step.

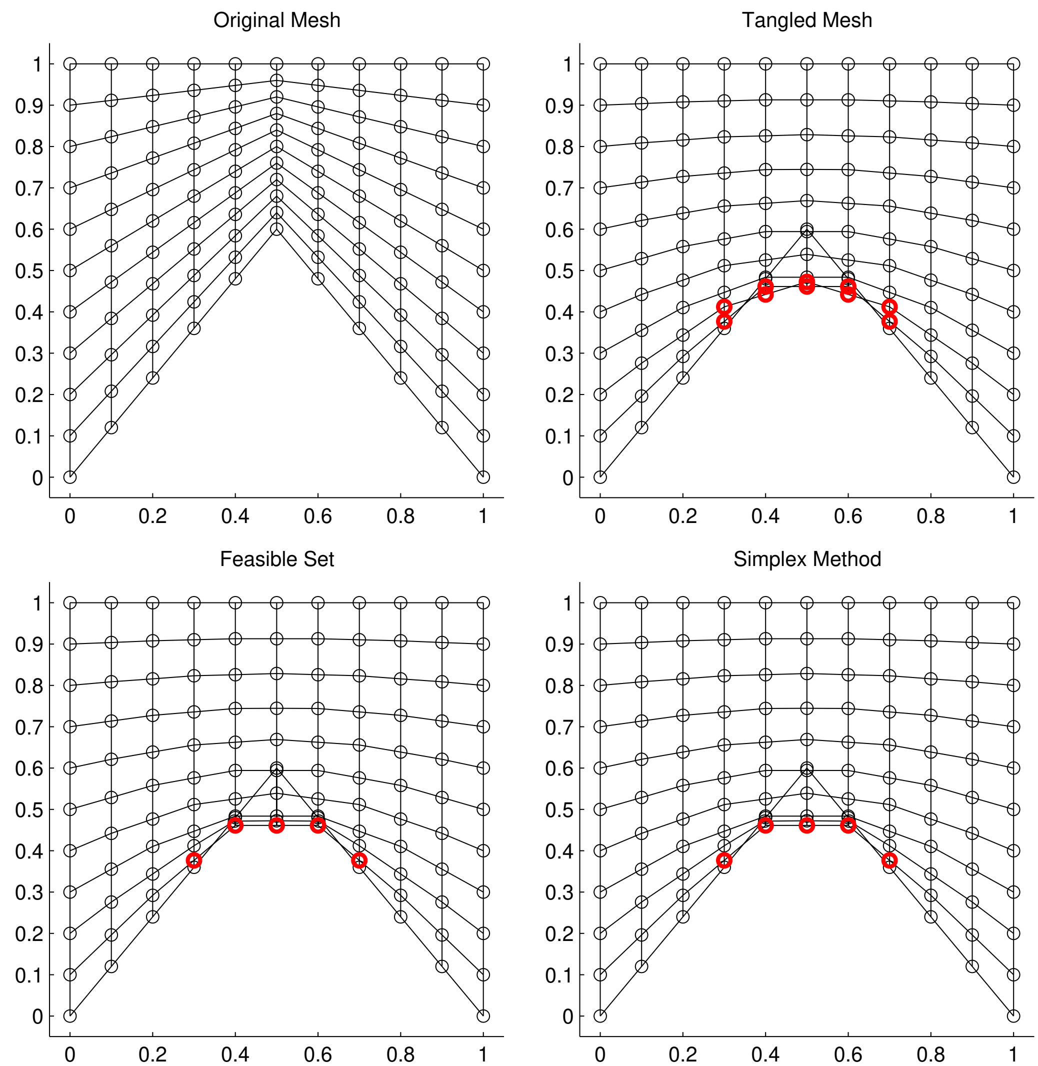

Figure 2.13: *Test 2D-4. Top: Original and tangled mesh. Bottom: results of feasible set - based approaches. Central vertices of remaining bad patches are marked red.*

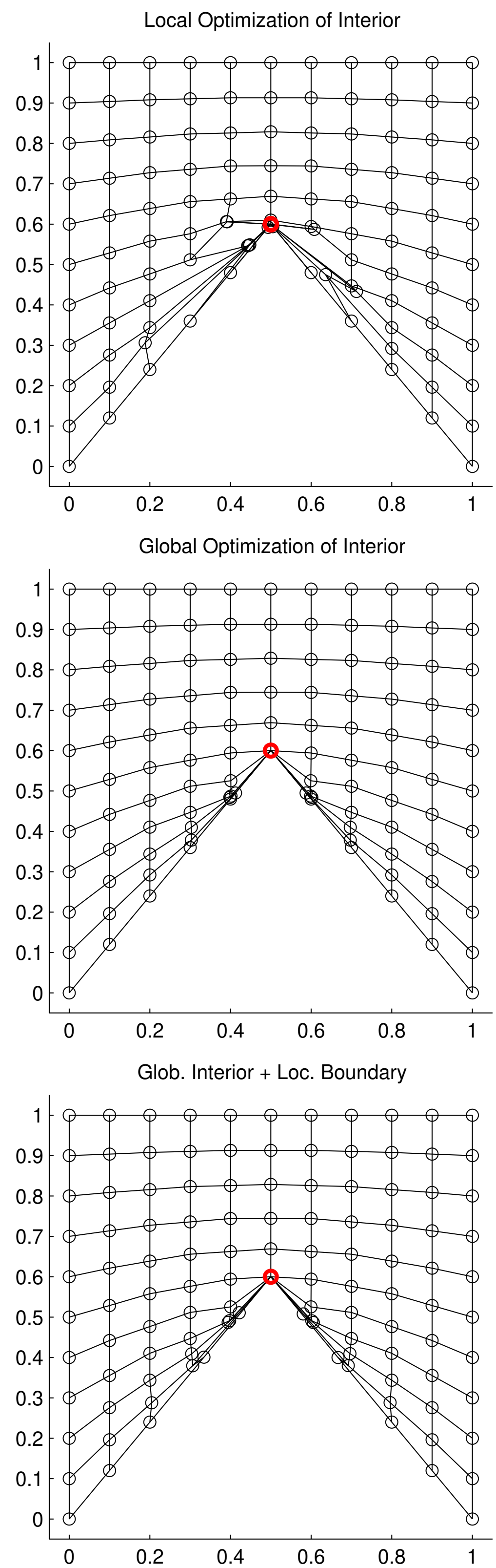

Figure 2.14: *Test 2D-4. Results of optimization approaches, moving interior and/or exterior vertices. Central vertices of remaining bad patches are marked red.*

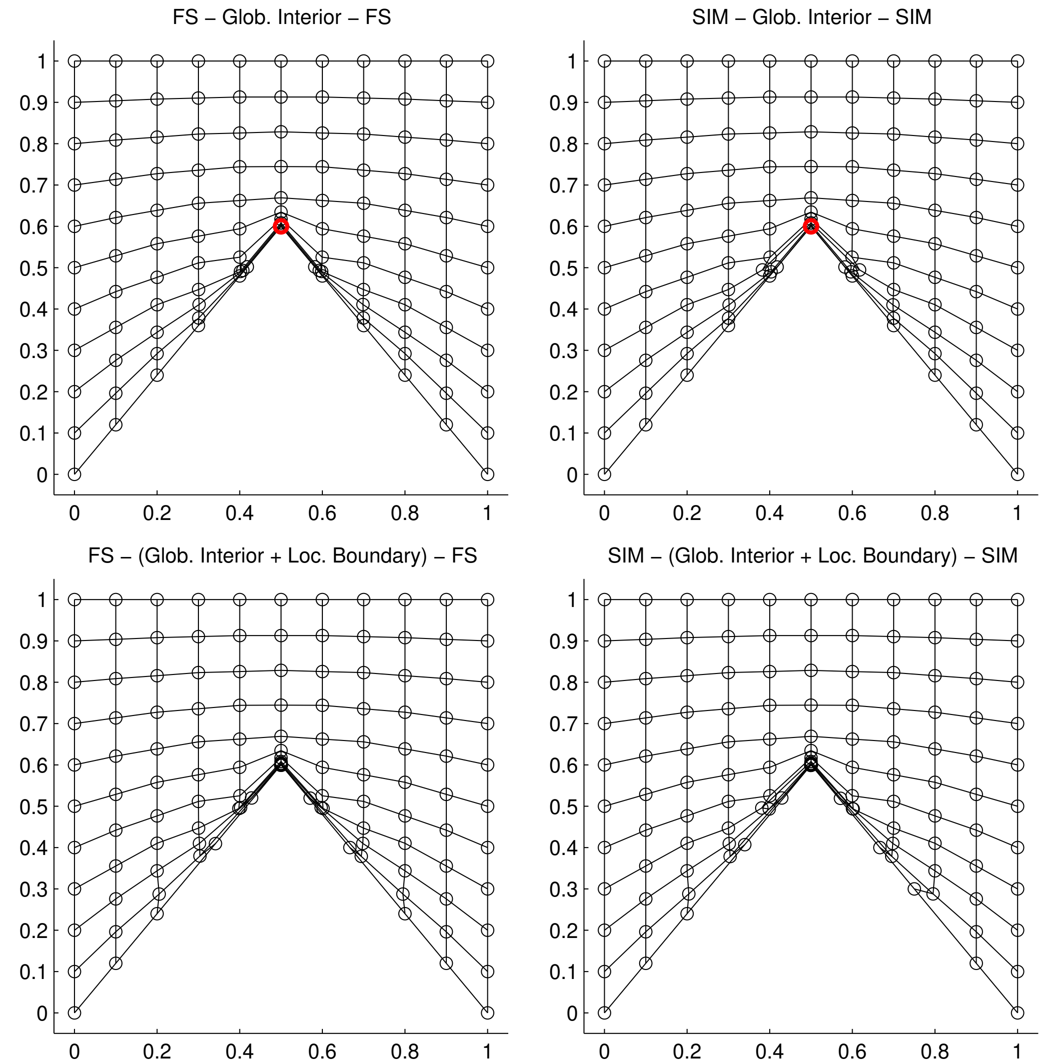

Figure 2.15: *Test 2D-4. Results of three-stage algorithms, using various combinations of basic methods. Central vertices of remaining bad patches are marked red.*

2.3.4 Randomly Tangled Texas (2D-5 and 2D-6)

In the last two examples, we took an Texas-shaped triangular mesh and tangled the interior vertices randomly.

In 2D-5, the maximum random perturbation of each vertex in each coordinate direction was set to 1/10 of the domain size (see Fig. 2.16). As can be seen in Fig. 2.17, all one-stage methods succeded to untangle the entire mesh, so we in fact did not need the three-stage approaches.

In 2D-6, the maximum allowed perturbation was twice as big, i.e. 1/5 of the domain in each direction (see Fig. 2.18). Results are shown in Fig. 2.19. For completeness, here are the numbers of still bad patches, i.e. patches still containing negative volume triangles (for feasible set - based methods) or zero volume triangles (for optimization): Feasible Set 17, Simplex Method 17, Local Optimization 8, Global Optimization 6, Three-stage algorithms 0.

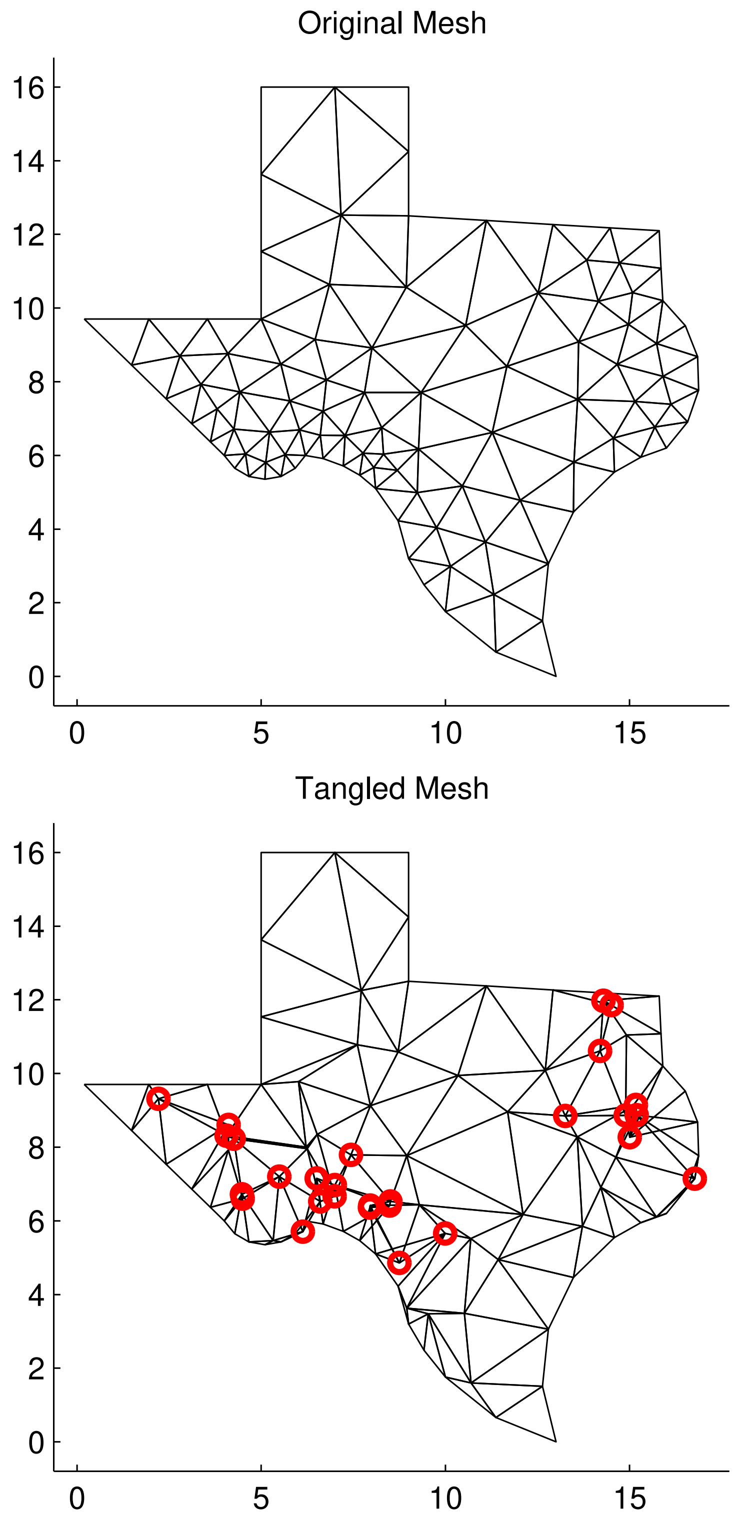

Figure 2.16: *Test 2D-5. Original and tangled Texas. Maximum perturbation* 1/10 *of the domain size in each coordinate direction. Central vertices of remaining bad patches are marked red.*

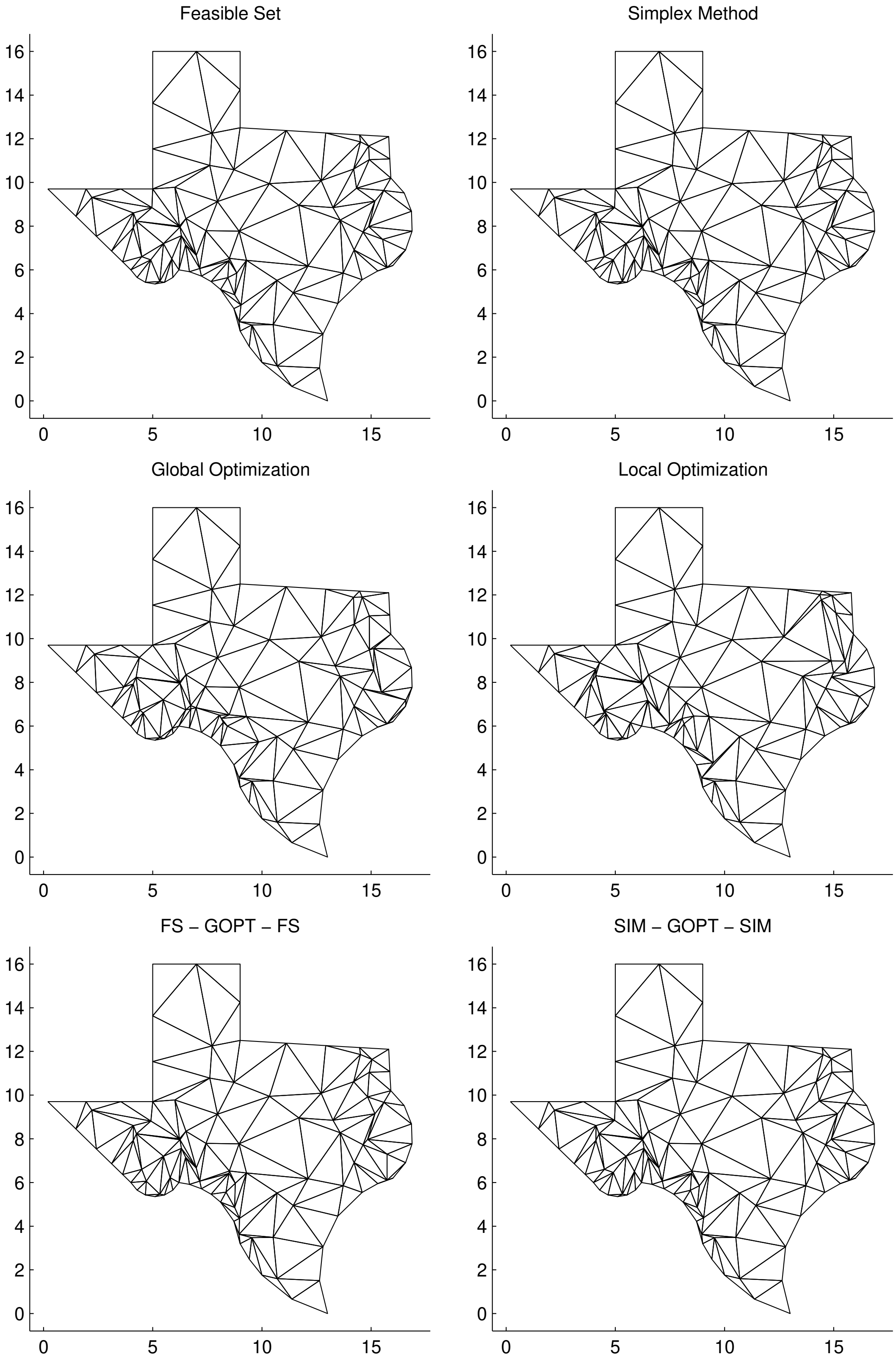

Figure 2.17: *Test 2D-5. Results of Feasible Set, Simplex Method, Local and Global Optimization and both Three-stage approaches.*

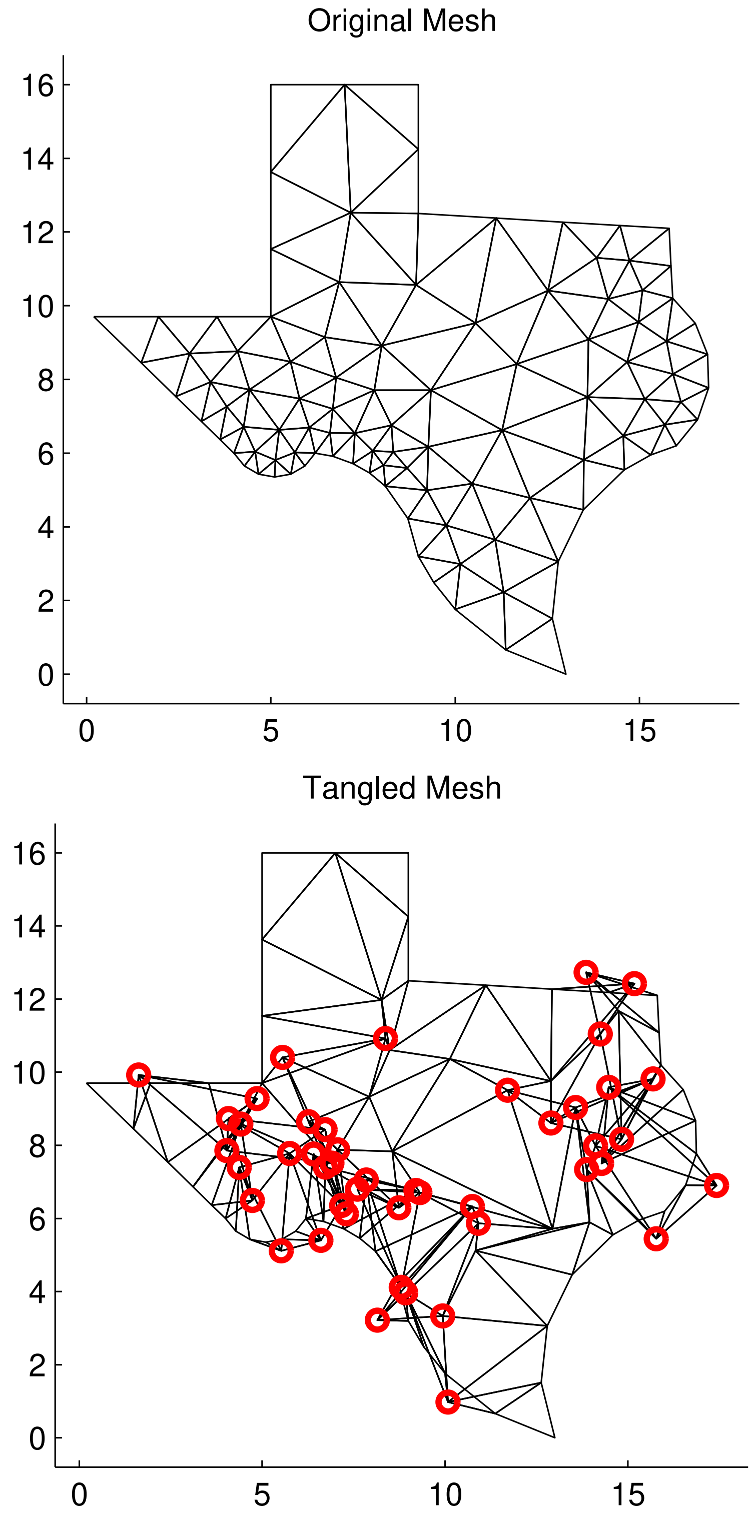

Figure 2.18: *Test 2D-6. Original and tangled Texas. Maximum perturbation* 1/5 *of the domain size in each coordinate direction.*

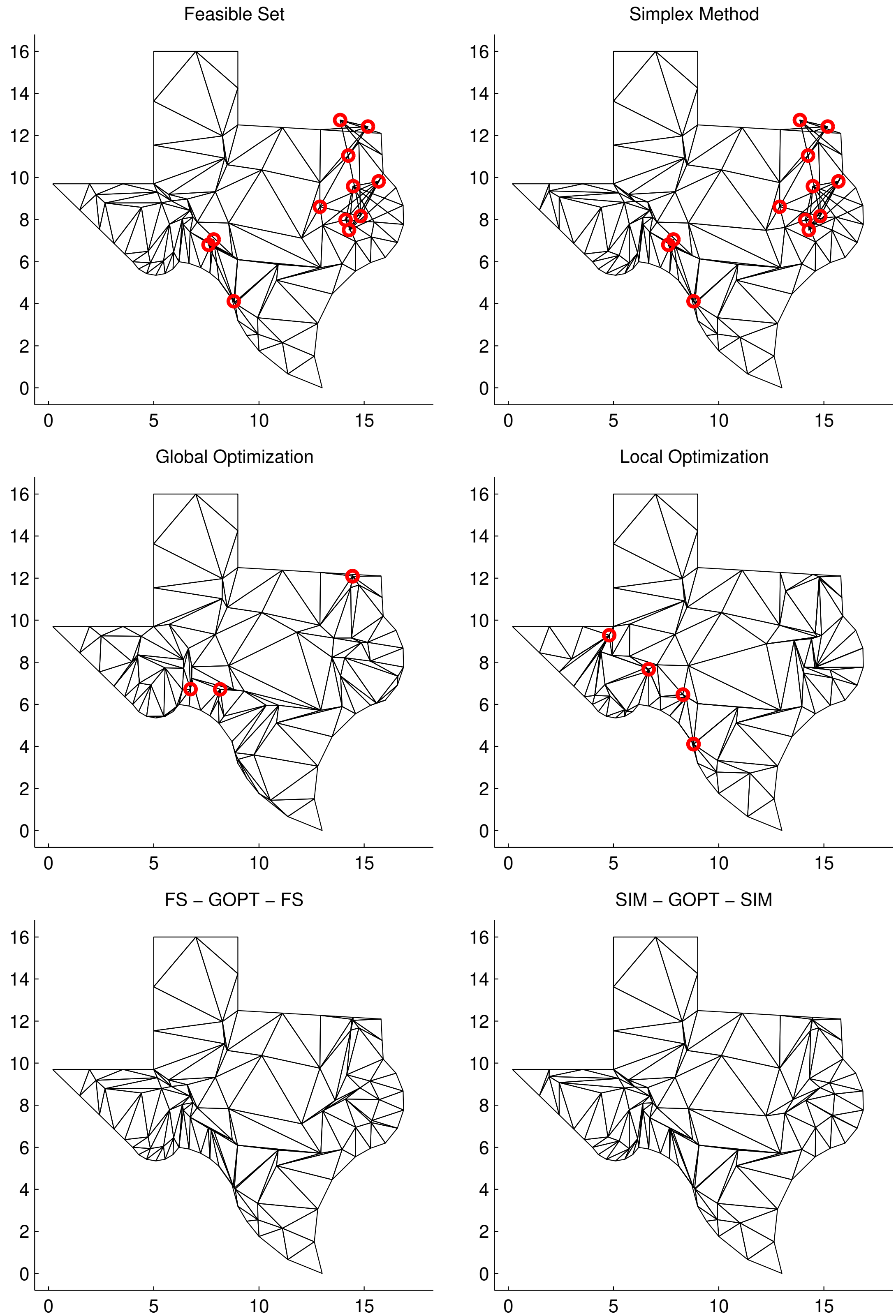

Figure 2.19: *Test 2D-6. Results of Feasible Set, Simplex Method, Local and Global Optimization and both Three-stage approaches. Central vertices of remaining bad patches are marked red.*

Chapter 3

3D

3.1 Description of Untangling Methods

The extension of our 2D methods to threee dimensions is straightforward. There are not any new ideas in 3D.

3.1.1 Feasible Set via Simplex Method (SIM)

Since the procedure for finding intersections of 3D bodies would be very time consuming, its use in our code would be inefficient. Thus we decided not to implement the approach, which finds analytically the entire feasible set. Instead, we use only the simplex method approach to untangle the patches. In complete analogy to 2D, we first find the constraints (here planes) imposed by surrounding vertices and then maximize functions $f = x$, $f = -x$, $f = y$, $f = -y$, $f = z$ and $f = -z$. If the feasible set is not empty, this finds at least its subset.

3.1.2 Local Optimization (LOPT)

As an objective function of a 3D patch, we take

$$f_\beta = \sum_{i=1}^{N_c} \left(|V_i - \beta| - (V_i - \beta)\right)^2 \tag{3.1}$$

where V_i are volumes of the connected tetrahedra, multiplied by two, and N_c their number (i.e. in a patch). The volumes can be computed by

$$V = \left(\vec{a} \times \vec{b}\right).\vec{c} \tag{3.2}$$

These are positive only if the tetrahedron is not inverted, i.e. if its vertices are ordered as follows: If P_1, P_2 and P_3 are arranged counterclockwise in the x-y plane, P_4 has to be in the halfspace, where z is positive (see Fig. 3.1)

Break conditions for local optimization are the same as in 2D.

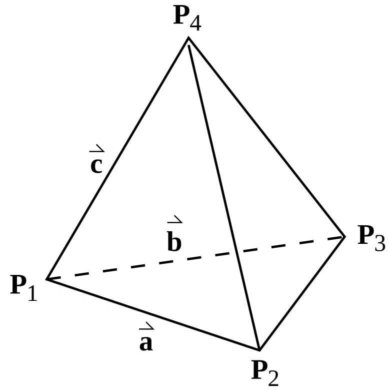

Figure 3.1: *Ordering of vertices in a valid tetrahedron.*

3.1.3 Global Optimization (GOPT)

Global objective function can be again considered as a sum of local objective functions over all patches. For efficiency, we again sum rather over tetrahedra, to take each of them only once.

3.1.4 Three - Stage Algorithm

As in 1D and 2D, a composite method, consisting of feasible set, global optimization and feasible set was implemented.

3.2 Tests

The results can be better seen in movies generated by package `unstruc3`.

Bibliography

[1] Knupp P.M.: *Hexahedral and Tetrahedral Mesh Untangling.* Engineering with Computers, vol. 17 (No. 3), pp. 261-268, 2001.

[2] Knupp P., Steinberg S.: *Fundamentals of Grid Generation.* CRC, Boca Raton, 1993.