A Compression Method for FEM Quadrilateral Data

SEBASTIAN KRIVOGRAD, MLADEN TRLEP, BORUT ŽALIK
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ulica 17, SI - 2000 Maribor
SLOVENIA

Abstract: - This paper presents a new approach for lossless compression of engineering data, represented by quadrilaterals. The presented method works in two steps. In the first step a topology is compressed and in the second step the entropy in the remaining data is removed. The topology compression is based on ten states; three of them are automatically recognized by the compressor/decompressor and the remaining are coded by four commands. Our topology compression algorithm has been compared with Isenburg algorithm [1] and it turned out as highly competitive. During the topology compression, the geometric data and application-specific engineering data are also prepared for the compression. The results of compression have been compared with popular PkZip, which is frequently used for lossless compression of FEM data. As shown in the paper, we achieved considerably better results.

Key-Words: - FEM quadrilateral mesh, compression, topology, geometry

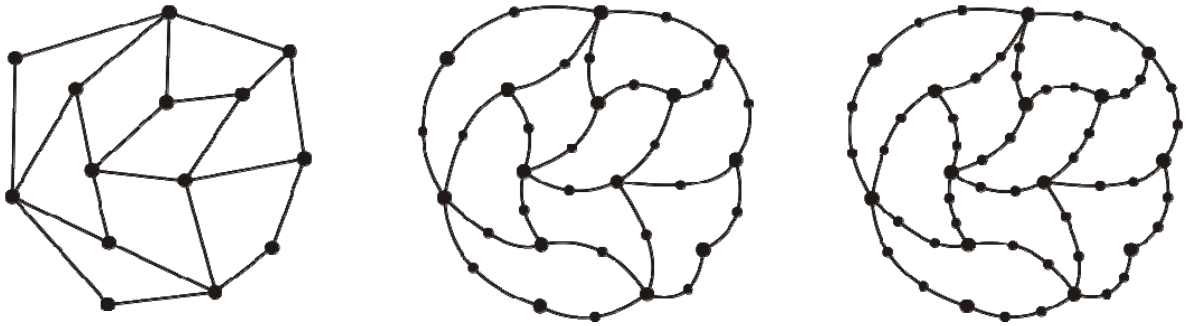
1 Introduction

The quadrilateral meshes are the second most usable meshes at representing the 3D object. They are also popular in the Finite Elements Method (FEM), which is the most popular and investigated method for various engineering and scientific calculations [2]. With the increase of computer power the meshes become larger what cause problem for storage and for transferring this data over internet. Both problems can be minimized with compression algorithms. They are a lot of different approaches how to minimize the used space [3], but those classical methods are not optimal for compression of arbitrary type of meshes. Therefore, the compression of meshes has become very hot topic [1,4-7]. All of those algorithms except [7] are optimized for visualization and not for engineering simulations.

In this paper we present a new method for lossless compression of quadrilateral meshes, as they appear in engineering (Electromagnetic FEM data were used for the case study). The presented method, contrary to others methods, accepts quadrilaterals of arbitrary orders (see Fig. 1) and efficiently handles engineering application-specific numerical information. Experiments have confirmed that the proposed method gives very promising results and that it is superior against general-purpose compression methods, which are usually applied in practice.

2 The Algorithm

The FEM data have three types of data; topological data (how vertices are linked together), geometrical data (coordinates of each vertex) and application-specific data associated with each vertex and



a) b) c)

Fig, 1: FEM data of a) first, b) second and c) third order

quadrilateral. The presented algorithm works in two steps:

1. Firstly, the topology is compressed. Simultaneously, geometric and application-specific data are properly arranged in the auxiliary lists.
2. Secondly, entropy coding of the data stored in the auxiliary lists is performed.

2.1 Topology compression

Each vertex in a quadrilateral mesh generally defines more quadrilaterals. The number of quadrilaterals determined by a vertex is considered a vertex degree. At the beginning, the quadrilateral mesh is analyzed, and the vertex degrees are determined. As our algorithm compresses only manifold quadrilateral meshes we first search for boundary edges (they determine only one quadrilateral) (see Fig. 2). If they are such edges, they are considered as *active edges* and stored in the list of active edges (*ListOfActiveEdges*). They can form one or more mutually non-connected loops (Fig. 2). If the mesh represents the closed surfaces, there are no bordering edges. In this case, an arbitrary quadrilateral is selected and extracted from the mesh. Its edges become members of the *ListOfActiveEdges*. The vertices defining the active edges are named *active vertices* and are placed in the *ActiveVerticesStructure*.

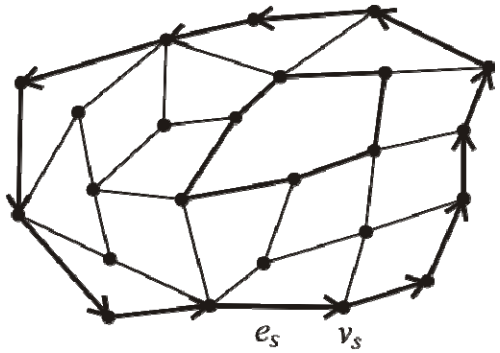


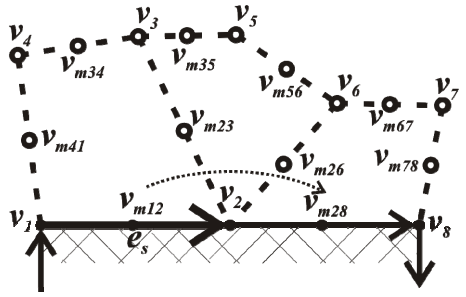
Fig. 2: Input manifold quadrilateral mesh

The next step of the compression process is selection of one of the active edges. It is named a *selected edge* e_s. The remaining active edges in the same loop are then oriented according to the first selected edge (Fig. 2). The end vertex of the selected edge is treated as a *selected vertex* v_s (Fig. 2). The configuration of the quadrilaterals around the e_s and v_s defines the characteristic states of the algorithm (the states of the algorithm will be explained later). The codes defining the states are stored in the list of commands (*ListOfCommands*). According to the states also the proper quadrilaterals are compressed and removed from the quadrilateral mesh. Their application-specific information's are stored in *ListOfQuadrilaterals*. After each step of compression the *ListOfActiveEdges* and *Active-VerticesStructure* are updated. The algorithm is terminated when the *ListOfActiveEdges* is empty.

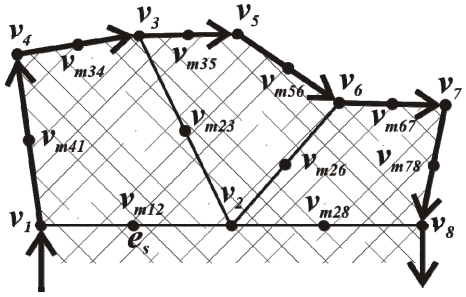
2.1.1 States of the compression process

The presented approach uses ten different states. Three of them are solved automatically, the rest are expressed by only four commands:

- **ADD.** This command is used when more than one quadrilateral originates in vertex v_s regarding the edge e_s (Fig. 3a). In this case we compress all uncompressed quadrilaterals surrounding vertex v_s in one step. In the situation shown in Fig. 3, command **ADD** inserts coordinates of the vertices and all application-specific information related to the vertices v_4, v_3, v_5, v_6, v_7 and v_8 into *ListOfVertices*, and their vertex degrees into *ListOfDegrees*. When there are quadrilaterals of second and higher orders then the inner vertices are inserted in the *ListOfVertices* in the order in which they are met. In Fig. 4 we have quadrilaterals of second order so additionally stored vertices are v_{m41}, v_{m34}, v_{m23}, v_{m35}, v_{m56}, v_{m26}, v_{m67}, and v_{m78}. In this way, the topological data (stored in *ListOfCommands*, *ListOfDegrees* and *ListOfNumbers*) do not change. At the end, the compressed quadrilaterals are marked as used and their application specific data are inserted



a)

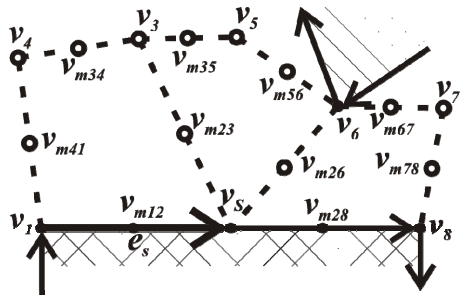


b)

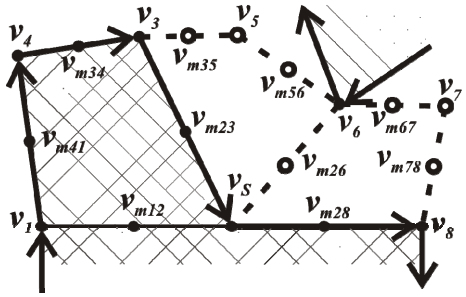
Fig. 3: Command **ADD**: beginning state (a), ending state (b)

into *ListOfQuadrilaterals*. The *ListOfActive-Edges* is updated to describe the situation shown in Fig. 3b. The command **ADD** is the most frequently used and all remaining commands just solve special cases.

- **ADD ONE.** It is used when more than two quadrilaterals originate in the selected vertex v_s, and any of the vertices around v_s has already been used, except the first one. In Fig. 4a, vertex v_6 has already been used and, therefore, the command **ADD** cannot be applied. In such a case, only the first quadrilateral determined by vertices v_4 and v_3 is compressed.



a)

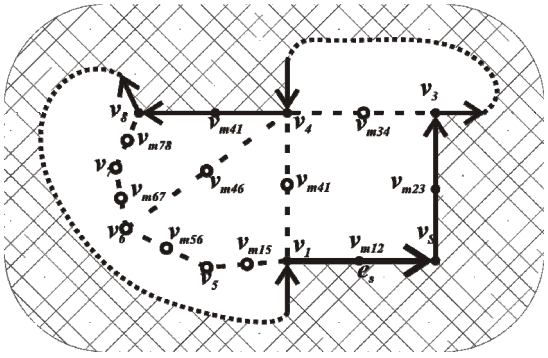


b)

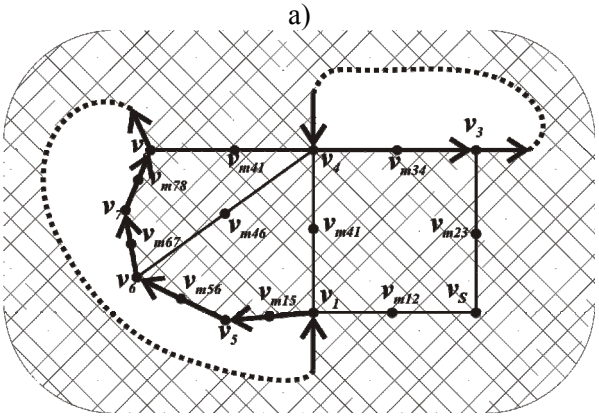
Fig. 4: Command **ADD ONE**: detection of the state (a), result (b)

- **SPLIT MERGE.** This command is applied in three main cases and each of them is divided into two parts (if the loop is going to be split or merge). The first case is shown by Fig. 5 and the second in Fig. 6:
  - The loop has to be split when:
    - one or more vertices of the first quadrilateral has already been used (vertex v_4 in Fig. 5a), and
    - v_4 is a member of the same loop as v_s.

    Because the decompressing algorithm does not know which vertex is vertex v_4, its variable index is inserted into *ListOf-Numbers*. The loop is split by inserting a chain of vertices (v_5, v_6 and v_7 in Fig. 5b). The number of vertices in the chain is inserted into *ListOfNumbers* and the information about the vertices into *ListOfVertices* and *ListOfDegrees*. In this

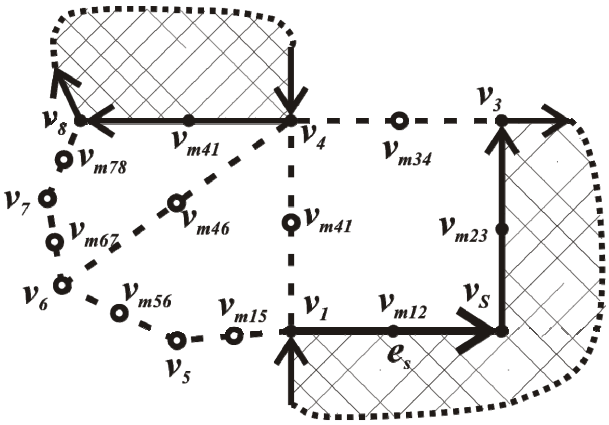


a)

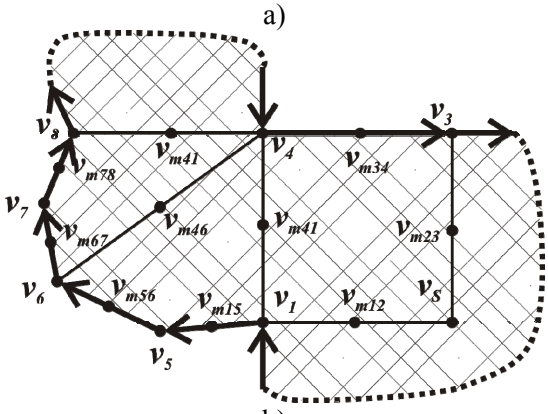


b)

Fig. 5: Split by command **SPLIT MERGE**: detection of the state (a), result (b)



a)



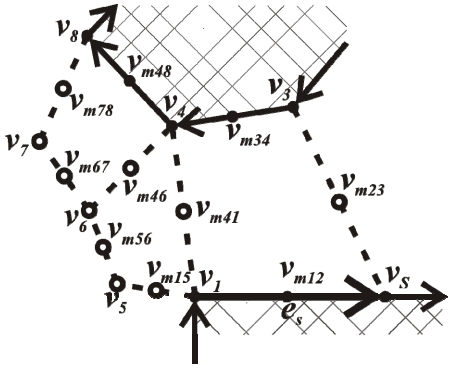
b)

Fig. 6: Merge by command **SPLIT MERGE**: detection of the state (a), result (b)

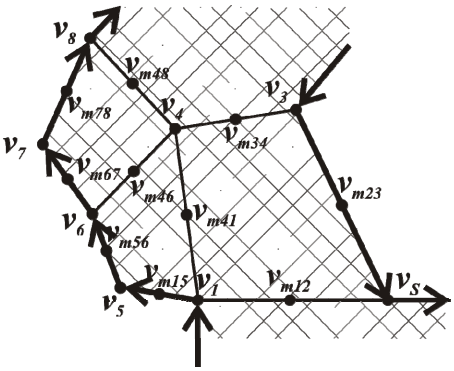
way additionally compressed quadrilaterals cause the division of the loop (see Fig. 5b). All information's about compressed quadrilaterals are inserted into *ListOf-Quadrilaterals* as they are processed.

- The equivalent situation is shown in Fig. 6, where two loops are merged. In this case v_4 is not a member of the same loop as v_s. Both loops have to be oriented in the same way.

As mentioned, Fig. 5 and Fig. 6 are showing only one situation where **SPLIT MERGE** command is applied. There are two more situations. To distinguish between them, one index is inserted into *ListOfNumbers* (*0* if situation shown in Fig. 5 or Fig. 6, *1* if situation shown in Fig. 7 and *2* if situation shown in Fig. 8).



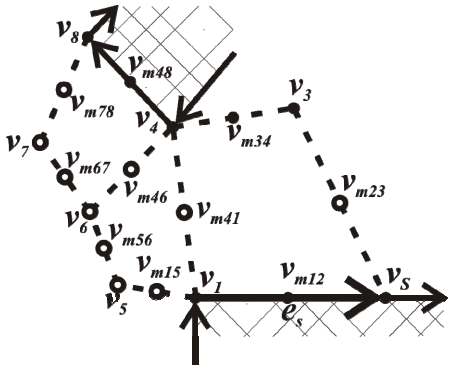
a)



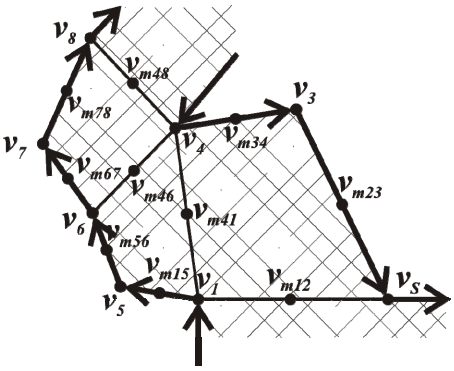
b)

Fig. 7: Second situation for command **SPLIT MERGE**: detection of the state (a), result (b)

- **SKIP.** This command is used in two situations. In first situation it is caused by the command **SPLIT MERGE**. Namely, it could happen that a vertex in the chain of vertices has already been used. In this case, the selected edge is left in *ListOfActiveEdges* and the next edge from this list is selected. Second situation is shown in Fig. 9. Also in this case command **SPLIT MERGE** could be used, but we could split/merge three different loops, what would cause a lot of problems to the control of compression process so command SKIP is used in such cases.



a)



b)

Fig. 8: Third situation for command **SPLIT MERGE**: detection of the state (a), result (b)

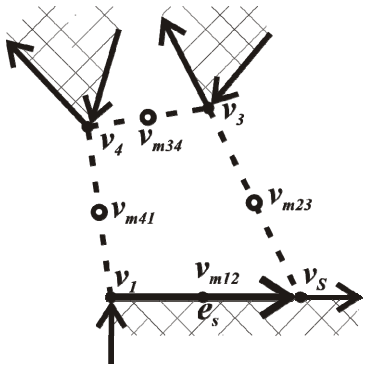
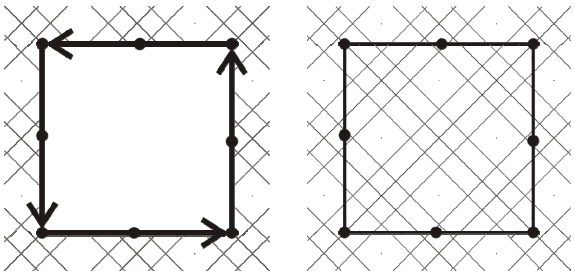


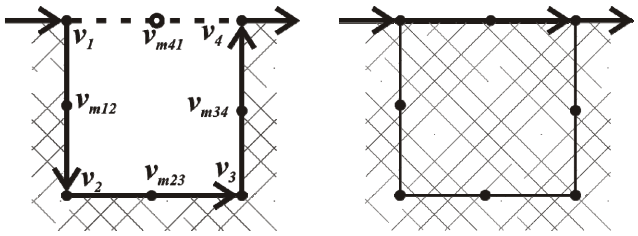
Fig. 9: Command **SKIP**

- The algorithm is able to automatically close the round around any vertex when its vertex degree number becomes 1. There are three possibilities where automatic close is applied:
  - When the compression of the loop is finished (Fig. 10). In this case only the information about compressed quadrilateral is inserted into *ListOfQuadrilaterals*.
  - The second situation is shown in Fig. 11. If the quadrilateral mesh is of higher order then the middle vertices are insterted into the *ListOfVertices* (v_{m41} in Fig. 11) else only the



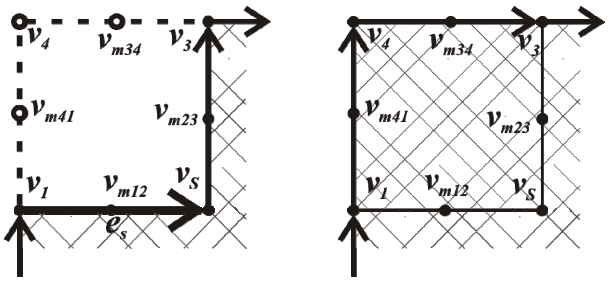
a) b)

Fig. 10: Automatic closure: detection of the state (a), result (b)



a) b)

Fig. 11: Automatic closure: detection of the state (a), result (b)



a) b)

Fig. 12: Automatic closure: detection of the state (a), result (b)

information about compressed quadrilateral is inserted into *ListOfQuadrilaterals*.

- The third situation (Fig. 12) is the most common situation for automatic close. In this situation v_4 is inserted into *ListOfVertices* and its degree number into *ListOfDegrees*.

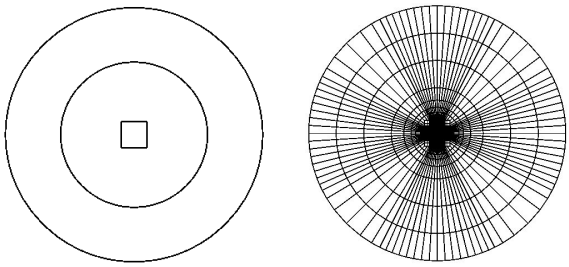
2.2 Entropy coding

For entropy coding the same algorithm as presented in [7] was used. First all floating point numbers were lossless transformed into integer numbers according to there hexadecimal representation and then all integer numbers were coded by Huffman coding, adaptive Huffman coding, arithmetic coding, or RLE [3].

3 Results

Two kinds of comparisons were made. First the efficiency of our topology compression algorithm has been compared against the Isenburg's algorithm [1] and second the comparison of compression of the complete data set of the engineering data (FEM data) was made against the popular and widely used *PkZip* package (WinZip 9.0 [9]). Isenburg's implementation uses lossy compression of geometric data so we could not use it [1].

For comparison of topology compression the engineering FEM data from electromagnetic have been used (Fig. 13). The total number of bytes needed for compressing the topology has been used for comparison. The proposed implementation is worse when number of vertices is less than 20,000. If there is more vertices then 20,000 then our methods becomes better than Isenburg algorithm (see Table 1).



FEM Model Quadrilateral FEM model

Fig. 13: Real FEM examples

However, topological data represent just a small part of data being compressed. Geometric data and engineering-specific data require much more space. In our case vertices and quadrilaterals are associated with the following data:

- vertices: geometric data (x and y coordinates

No. of vertices	No. of quadrilaterals	Isenburg		Our method		% of Isenburg
		B	b/vertex	B	b/vertex	
1,145	1,100	16	0.11179	22	0.15371	137.50
1,921	1,860	18	0.07496	22	0.09162	122.22
12,041	11,868	23	0.01528	24	0.01595	104.35
22,681	22,428	25	0.00882	24	0.00847	96.00
52,009	51,876	26	0.00400	25	0.00385	96.15
99,529	99,396	28	0.00225	25	0.00201	89.29

Table 1: Comparison of the total number of bytes needed for compression of all topological data

No. of vertices	No. of quadrilaterals	Order	Sizes [B]		
			DAT	DAT.ZIP	CDAT
12,041	11,868	1	2,422,994	214,770	21,909
36,049	11,868	2	5,035,545	476,325	47,642
60,057	11,868	3	10,474,324	1,064,122	104,254
52,009	51,876	1	9,918,946	965,189	163,683
155,893	51,876	2	20,615,842	2,156,764	335,437
259,777	51,876	3	43,054,324	4,563,352	673,235
99,529	99,396	1	18,995,266	1,845,214	323,149
298,453	99,396	2	39,481,282	4,132,876	632,156
497,377	99,396	3	82,246,354	8,535,733	1,336,463

Table 2: Comparison of total sizes of FEM data

represented by floating-point numbers), value of unknown function of electric or magnetic potential in this vertex (floating-point number), and type of boundary condition (integer number).

- quadrilaterals: indices of vertices defining the quadrilateral (topology information compressed according to the description in Section 2), type of material covered by this quadrilateral (integer number), property of the used material (floating-point number) and source-value of the electro-magnetic field (two floating-point numbers).

The results are summarized in Table 2, where **DAT** represents the original size of the input data stored in ASCII file, **DAT.ZIP** means the size of compressed input file with *PkZip*, and **CDAT** represents the compressed input data using the proposed method.

As can be seen in Table 2, the proposed method is very efficient as the compressed file is less than 2% of the original data stored in ASCII. It is also much better than *PkZip*. Our compressed file occupies around 15% of original data compressed by *PkZip*.

4 Conclusion

This paper introduces a new approach for compression of quadrilateral engineering meshes. In this case vertices and quadrilaterals carry additional engineering information. In addition, quadrilaterals can have different orders. The topology compression was compared with the Isenburg's approach [1]. The presented algorithm gives better results if number of vertices is bigger then 20,000.

The aim of this work was to develop a method suitable for engineering applications. As shown by the experiments, this proposed method achieves encouraging results. It compresses the input ASCII file describing the quadrilateral mesh equipped with engineering data to 2% of the initial data size. Comparison with popular *PkZip* was also carried out. The proposed method occupies only around 15% of the input ASCII compressed by *PkZip*.

Acknowledgements:
We are grateful to the Slovenian Research Agency for supporting this research within the project Z2-6661-0769-04/2.12 - Compression of elements appearing in the final elements methods (FEM) and within the project P2-0041 – Computer systems, methodologies and intelligent services.

References:

[1] Isenburg M., Compressing Polygon Mesh Connectivity with Degree Duality Prediction. *In: Graphics Interface'02 Conference Proceedings*, 2002, pp. 161-170.

[2] Trlep M., Hribernik B., Unified Approach to Solving a Steady-state Electromagnetic Field. *IEEE Transactions on Magnetics*, Vol.33, No.2, 1997, pp. 1974-1977.

[3] Salomon D., *Data Compression: The Complete Reference*. Springer-Verlag, New York, 1997.

[4] Turan G., On the Succinct Representation Of Graphs. *Discrete Applied Mathematics*, Vol.8, No.3, 1984, pp. 289-294.

[5] Touma C., Gotsman C., Triangle Mesh Compression. *Graphics Interface'98 Conference Proceedings*, 1998, pp. 26-34.

[6] King D., Rossignac J. and Szymczak A., Connectivity compression for irregular quadrilateral meshes. Georgia Tech, TR–99–36.

[7] Krivograd S., Trlep M, Žalik B.,. A compression method for engineering data represented by triangular meshes. WSEAS Transactions on Mathematics., July 2005, vol. 4, iss. 3, pp. 266-272.

[8] http://www.winzip.com.