POLLACK PERIODICA
An International Journal for Engineering and Information Sciences
DOI: 10.1556/606.2016.11.3.15
Vol. 11, No. 3, pp. 165–176 (2016)
www.akademiai.com

APPROXIMATION METHODS FOR POST-PROCESSING OF LARGE DATA FROM THE FINITE ELEMENT ANALYSIS

[1]Štěpán BENEŠ, [2]Jaroslav KRUIS

Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7, Prague, 166 29, Czech Republic
e-mail: [1]stepan.benes@fsv.cvut.cz, [2]jk@cml.fsv.cvut.cz

Received 1 January 2016; accepted 28 May 2016

Abstract: The paper describes efficient methods to post-process results from the finite element analysis. Amount of data produced by the complex analysis is enormous. However, computer performance and memory are limited and commonly-used software tools do not provide ways to post-process data easily. Therefore, some sort of simplification of data has to be used to lower memory consumption and accelerate data loading. This article describes a procedure that replaces discrete values with a set of continuous functions. Each approximation function can be represented by a small number of parameters that are able to describe the character of resulting data closely enough.

Keywords: Finite element method, Finite element mesh, Data visualization, Approximation, Post-processor

1. Introduction

Finite Element Method (FEM), as well as other numerical methods used in scientific and engineering problems, generate large amount of data results in the form of numbers in output files. Numbers has to be translated to graphical representation to be suitable for humans to post-process. However, complex finite element analyses can generate output files with the size in order of gigabytes per each time step. As an example can serve thermo-hydro-mechanical analysis described in [1]. This analysis uses 6 unknowns per node, which leads to a large system of equations, high computational time and large amount of output data. These types of analyses are often parallelized on some supercomputer to reach the end of analysis in reasonable time. But the results are

then post-processed on commonly-used personal computer. Therefore, some kind of simplification and compression of results is needed to allow efficient post-processing. Data structures used for representation of large finite element meshes are described in [2]. Some techniques for compression of polygonal meshes can be found in [3], [4]. However, these methods are not suitable for handling the results from the finite element analysis.

Therefore, it was decided to develop a new compression method for visualization of the results from the finite element analysis. The main idea behind this method is to decompose discrete data in the problem domain into multiple levels of detail and then on each level of detail to describe the shape of discrete data by replacing them by continuous functions. Inspiration for this work is the Multigrid method [5], [6] and also some image reconstruction techniques [7]. The main task is to preserve the quality and accuracy of visualization using continuous functions that are as simple as possible (describable by a few parameters). Results of this compression method are presented in this paper.

Post-processor is required to display various kinds of data, e.g. temperature, displacements, stress, strain. These quantities are scalars, vectors or tensors of second order. Components of vectors and tensors could be considered in post-processing as a scalar and therefore scalars will be dealt in the following text. Every scalar is represented in the finite element analysis by a set of discrete values computed in nodes or Gauss points. In the following text, the set of discrete values describing a scalar will be denoted as the discrete function or original function, but approximation of the discrete values for graphical purposes by continuous function will be called approximation function (shape functions used in FEM are not used here).

2. Implementation

Results from the finite element analysis are discrete values. But it cannot be blindly replaced by a single approximation continuous function, because to achieve low approximation error, the approximation function would have to be very complex and would not save much memory to represent it. However, in the typical case there are areas in the results that have smooth function development and also areas where the function rapidly changes its character. It is therefore necessary to properly divide the domain to segments and apply approximation function in each segment separately.

The use of suitable division of function domain can lead to significantly better approximation with very simple continuous functions. But in general it is not possible to know the exact division of space in advance.

To automate this process the octree data structure can be used (see *Fig. 1*). It is hierarchical data structure that is often used in computer graphics. Its purpose is to recursively divide the space into octants (for 3D domains).

2.1. Octree generation

The algorithm for generating domain decomposition data structure in the form of octal tree is depicted in *Fig. 2*. The process starts with inserting all data into octree root

node, which is one big cube that surrounds (or contains) the whole mesh. Then the algorithm tries to replace the data by a single continuous approximation function. Then the approximation error is calculated and compared to some designated epsilon value that describes the required precision. If the error is too high, the algorithm continues, divides the cube into eight segments (smaller cubes) and sends the approximation errors for each data point to sub-nodes. Then the algorithm does the same thing in each sub-node. It continues until approximation is good enough in all segments. Sending approximation errors instead of original data is important, because it allows catching the low frequency character of function on top levels and high-frequency changes in function on bottom levels of octree. When the algorithm finishes, all original discrete data can be deleted, because the created octree data structure with approximation functions in its nodes is all that is needed to reconstruct the original data.

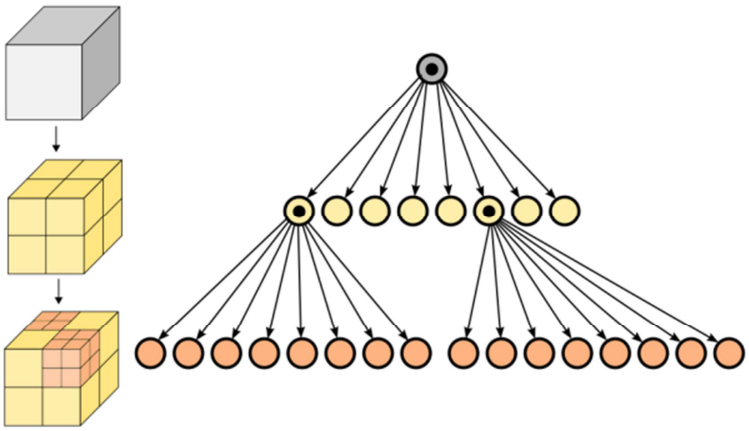

Fig. 1. Octree data structure visualization; left: recursive subdivision of a cube into octants; right: the corresponding octree

This approach was chosen to enable the top levels of the octree to describe main character of function (lower frequencies) and bottom levels of the octree catch higher frequencies of function values.

Computation of data value in some position is then made by traversing the octree from the root to the leaves. In each octree level, the coordinates of data point are put into approximation function and it yields the approximation error in current octree level. When all these values across all levels are summarized from the top to the bottom the original data value in the point is reconstructed.

2.2. Approximation functions

Various types of approximation functions were investigated and tested. Besides polynomial functions also Discrete cosine transform [8] and Wavelet transform [9] were examined. Since the data compression algorithm has to be very fast, simple polynomial approximation functions were preferred. Tri-linear interpolation function was chosen as

the best compromise between low approximation error and memory consumption. It needs 8 parameters to describe function shape. It is consistent with neighboring octree cells, almost 'seamless' transitions between octree cells.

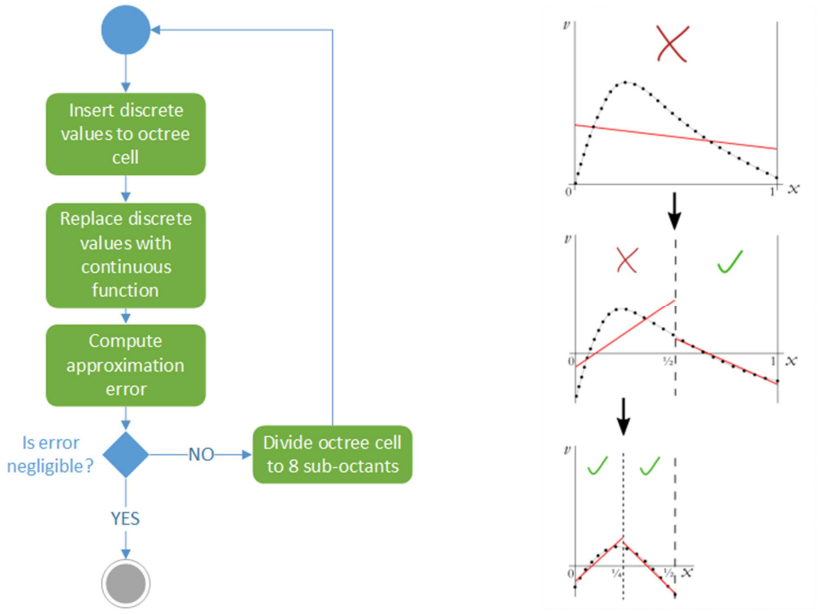

Fig. 2. Octree generation; a) activity diagram; b) example of octree creation driven by approximation error

Value v in the point with coordinates x, y, z is computed using tri-linear interpolation function in the form

$$v = c_1xyz + c_2xy + c_3xz + c_4yz + c_5x + c_6y + c_7z + c_8 . \tag{1}$$

The least squares method is applied to find parameters $c_1,\cdots,c_8$. The problem is solved by minimizing the sum of squared residuals G of the linear regression model

$$G = \sum_{i=1}^{N}(v_i - (c_1x_iy_iz_i + c_2x_iy_i + c_3x_iz_i + c_4y_iz_i + c_5x_i + c_6y_i + c_7z_i + c_8))^2 , \tag{2}$$

where N is number of values, which are interpolated. When the parameters of interpolation are known, value in any point of the approximated volume can be found simply by providing x, y and z coordinates of the point in the equation.

3. Results

The benchmark is designed to compare maximal relative approximation error, average error and compression ratio using tri-linear approximation functions. Maximal relative approximation error is the highest relative error of an approximation method in single element node across all data components and time steps. Average relative error is a weighted sum of relative approximation errors in all nodes and data components divided by the number of these approximations and is calculated as

$$e_{avg} = \frac{\sum_{i=1}^{n} \frac{|u_i - \hat{u}_i|}{u_{\max} - u_{\min}}}{\sum_{i=1}^{n} w_i}, \tag{3}$$

where n is the number of data points in original discrete function representing exact data values; x_i is the i-th value of original discrete function; $\hat{x}_i$ is the value of the approximation function in the same location; and w_i is the weight of the i-th test point with the meaning of volume surrounding the point. u_{max} and u_{min} are maximum and minimum values of original discrete function.

Compression ratio is memory consumption of the proposed data representation divided by memory consumption of original post-processor that does not use any data approximation techniques.

As a test analysis serve the thermo-mechanical analysis of model of Charles Bridge in Prague. Analysis results contain displacement vector values with three components (u, v and w) (see *Fig. 3*) and scalar values of temperature distribution (see *Fig. 4*). Analysis has 46 time steps. Total number of data sets that are processed by compression algorithm is therefore 184. Each data set has 73749 values that correspond to number of nodes in the finite element mesh. Results are summarized in *Table I*.

Table I

Benchmark results

	Max error [%]	Average error [%]	Compression ratio [%]
Displacement *u*	26.49	0.13	13.3
Displacement *v*	70.71	0.19	27.4
Displacement *w*	48.49	0.14	16.3
Temperature	*92.46*	0.41	59.0
Average		0.22	29.0

Fig. 5 contains visualization of the exact data values whereas *Fig. 6* contains visualization of approximation of the same data series. Significant visible

approximation errors are marked by arrows. Visualization of approximation is shown in *Fig. 7* where differential function of exact and approximated values are presented. Colors are emphasized for clarity. *Fig. 8* shows same kind of error on different data set. For better illustration of the method, the octree segments are also visualized using lines representing surrounding boxes of each octree segment.

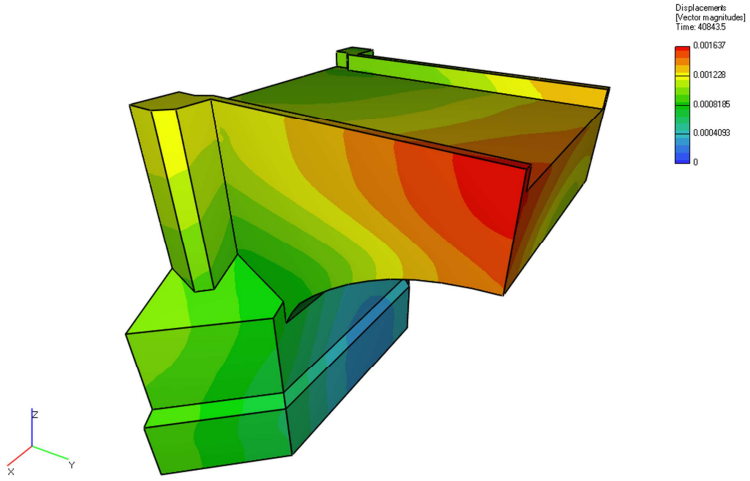

Fig. 3. Model of Charles Bridge in Prague; heat transport analysis results (displacements)

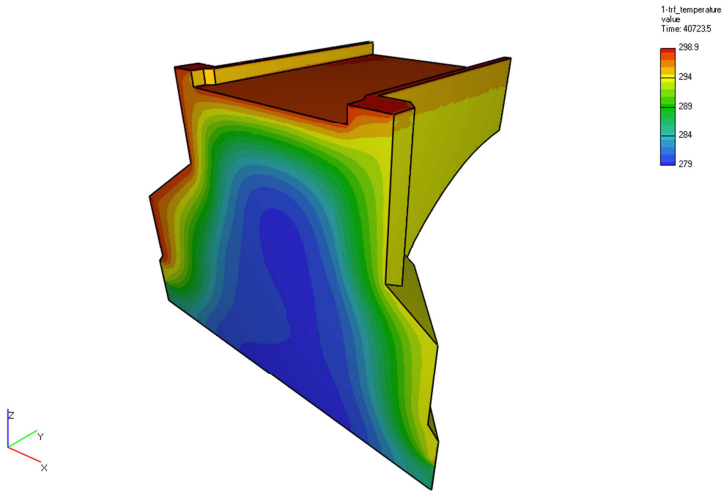

Fig. 4. Model of Charles Bridge in Prague; heat transport analysis results (temperature)

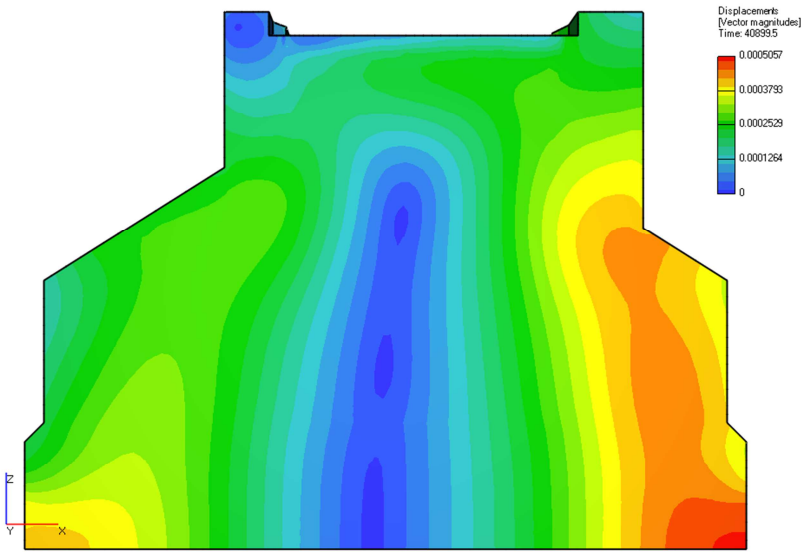

Fig. 5. Exact data values; no approximation applied

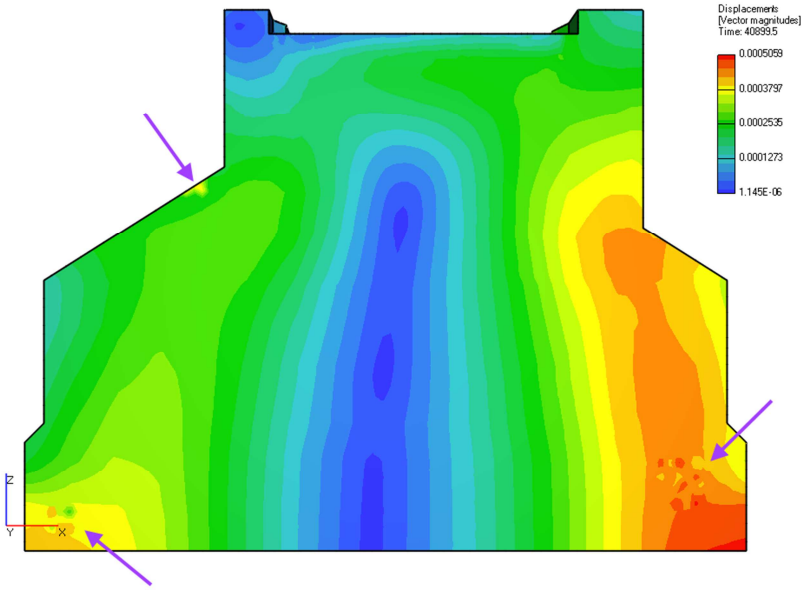

Fig. 6. Approximation method's artifacts (marked by arrows)

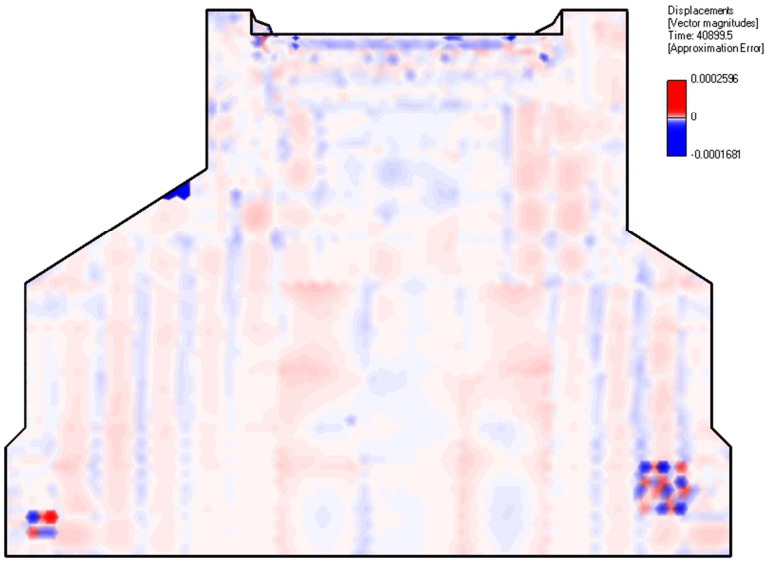

Fig. 7. Visualization of approximation error. Differential function between exact and approximation function

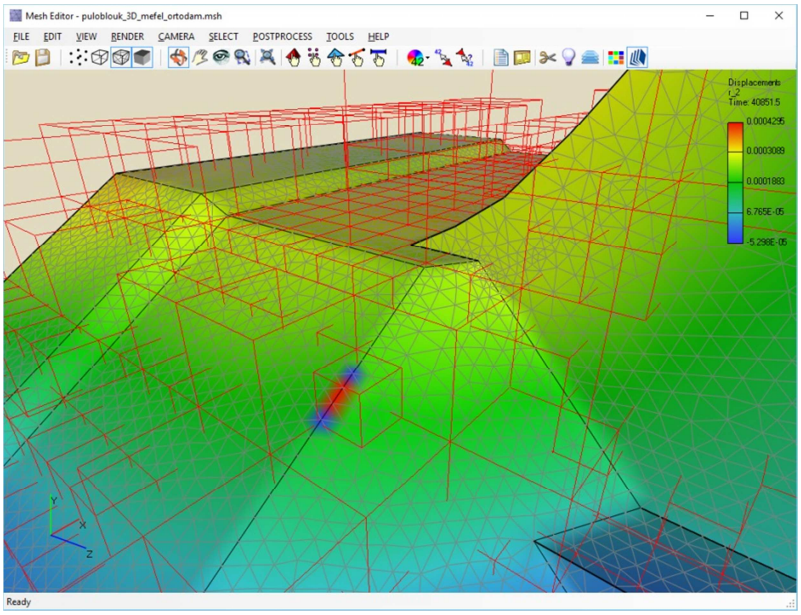

Fig. 8. Glitches in approximation function caused by octree-based space decomposition. Octree segments are visualized using their surrounding boxes

Fig. 9 shows results from the analysis of coupled hydro-mechanical behavior of soils that is described in detail in [10] and [11]. In *Fig. 10* the tri-linear approximation function is applied.

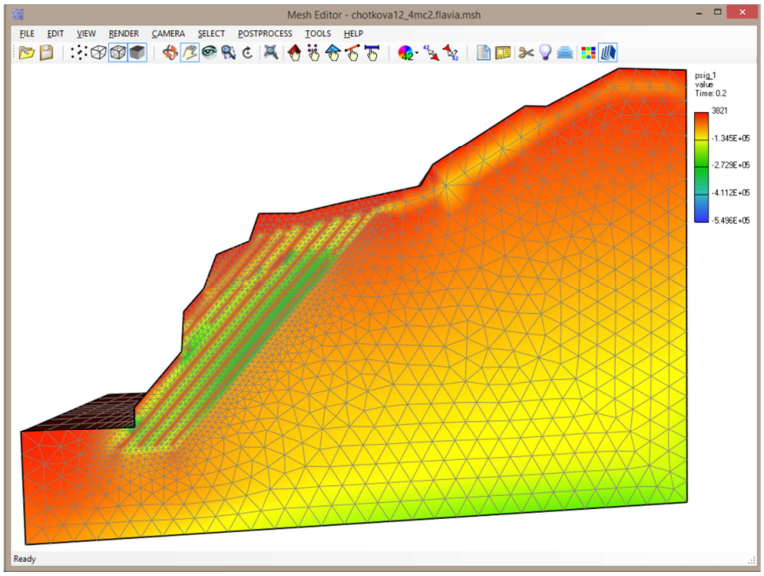

Fig. 9. Chotkova example. Exact data values, no approximation applied

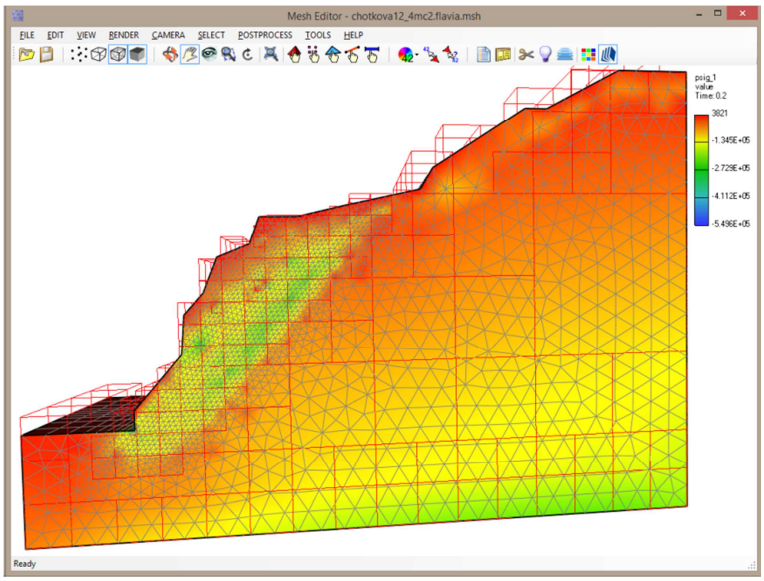

Fig. 10. Chotkova example. Tri-linear approximation of data values with visualization of octree segments using border lines

The octree-based compression algorithm preserves almost seamless transition between octree cells with varying depth of the octree according to smoothness of the approximated function. However, the method can't capture high frequent changes in data which is common problem of all lossy compressions. Differential function in *Fig. 11* illustrates the results presented in *Table I* by showing locations with maximal approximation error.

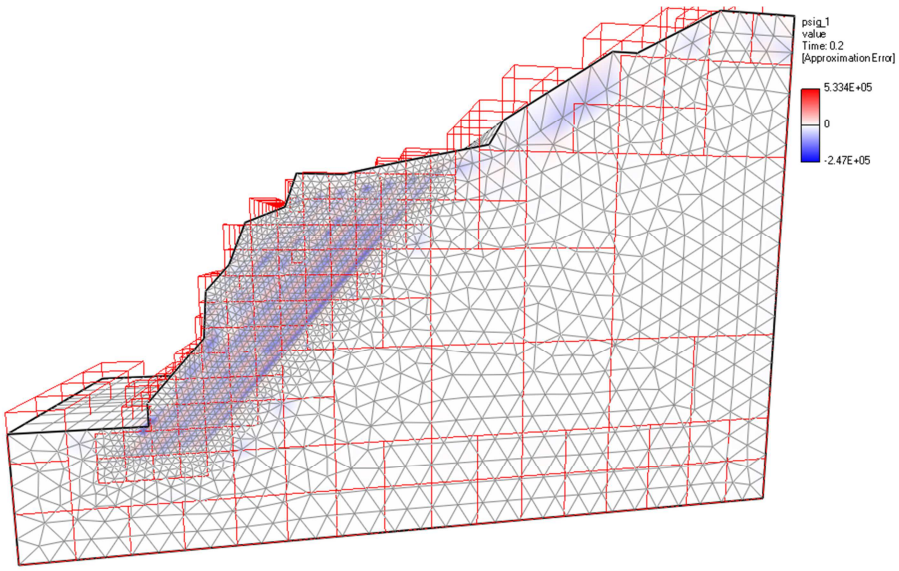

Fig. 11. Chotkova example. Visualization of approximation error. Differential function between exact and approximated data values. The octree segments are visualized using border lines

The imperfections of the method can have two causes. Artifacts can appear typically for results with high-frequency changes in data, for which the octree data structure cannot be fine enough. Special condition in the octree creation algorithm specifies the minimum number of discrete function values to be contained in the octree cell to replace them by the continuous approximation function. This minimum number is related to the type and order of the approximation method that is used in the algorithm. If the condition is not met, the octree cell cannot be divided into eight child cells even if the approximation error is still too high. The possible solution could be to allow the use of higher-order approximation functions in these rare cases, but it would grow the memory consumption and considerably complicate the algorithm.

The second reason for these kinds of errors is strictly local nature of the approximation algorithm that cares only about data values in current octree cell and does not take the neighbor segments into account. The transitions between cells can be sometimes far from smooth as can be seen in *Fig. 6* and *Fig. 7*.

4. Conclusion

A lossy compression method was developed for post-processing large amount of data obtained by complex finite element analyses. In average case the compression ratio is about 30% with quite low relative approximation error at 0.2%. However, in some extreme cases (rough, unpredictable function shape) the maximal error can be quite high, up to 100% and the method does not even guarantee any upper limit on the approximation error.

The compression ratio is not as low as was expected at the beginning, but in the following work the compression ratio can be significantly decreased by applying the same approach also for time - the temporal dimension of the problem. The current algorithm produces the continuous approximation functions for each time step. The algorithm can be extended to recognize the time steps in which the function does not change or changes linearly and can be therefore interpolated from neighboring time steps. Whole approximation functions in these steps can be then disposed, because they are not necessary - they can be computed from other time steps.

However, isolated but unpredictable and excessive maximum approximation error is the crucial disadvantage of the method. The future work will address these issues. Some possible direction can be to try more sophisticated types of approximation functions or some time-frequency transformations that will better describe discontinuities in input data e.g. Discrete cosine transform or Wavelet transform. Other inspiration for the work is the Multigrid method that can be used in the finite element method to accelerate iterative solvers by using levels of different mesh densities. Similar approach could be used for visualization of the results from the finite element method.

Acknowledgement

Financial support for this work was provided by project number 15-05935S of Czech Science Foundation. The financial support is gratefully acknowledged.

References

[1] Krejčí T., Koudelka T., Kruis J. Numerical modeling of coupled hydro-thermo-mechanical behavior of concrete structures, *Pollack Periodica,* Vol. 10, No. 1, 2015, pp. 19-30.

[2] Beneš Š., Kruis J. Efficient methods to visualize finite element meshes, *Advances in Engineering Software*, Vol. 79, 2015, pp. 81–90.

[3] Maglo A., Courbet C., Alliez P., Hudelot C. Progressive compression of manifold polygon meshes, *Computers & Graphics*, Vol. 36, No. 5, 2012, pp. 349–359.

[4] Váša L. Optimised mesh traversal for dynamic mesh compression, *Graphical Models*, Vol. 73, No. 5, 2011, pp. 218–230.

[5] Shaidurov V. V. *Multigrid methods for finite elements,* Mathematics and its Applications, Springer, 1995.

[6] Hackbusch W. *Multigrid methods and applications,* Springer. 2010.

[7] Magoulès F., Gbikpi-Benissan G. Coarse space construction based on Chebyshev polynomials for graphic analysis, *Pollack Periodica,* Vol. 9, No. 2, 2014, pp. 3-14.

[8] Roma N., Sousa L. A tutorial overview on the properties of the discrete cosine transform for encoded image and video processing, *Signal Processing*, Vol. 91, 2011, pp. 2443–2464.
[9] Li B., Chen X. Wavelet-based numerical analysis: a review and classification, *Finite Elements in Analysis and Design*, Vol. 81, 2014, pp. 14–31.
[10] Koudelka T., Krejčí T., Brouček M. Numerical modeling of consolidation processes under the water level elevation changes, *Advances in Engineering Software,* Vol. 62-63, No. 72, 2014, p. 166–178.
[11] Koudelka T., Krejčí T., Brouček M. Modeling of coupled hydro-mechanical problem for porous media, *11th International Conference of Numerical Analysis and Applied Mathematics*, 2013, American Institute of Physics Conference proceedings, Vol. 1558, New York, 2013, pp. 984–987.