Advances in Engineering Software 117 (2018) 8–17

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research Paper

Singular Value Decomposition used for compression of results from the Finite Element Method

Štěpán Beneš, Jaroslav Kruis*

Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague Thákurova 7, Prague 166 29, Czech Republic

ARTICLE INFO

Keywords:
Finite Element Method (FEM)
Compression
Singular Value Decomposition (SVD)
Randomized SVD
Data visualization
Data post-processing

ABSTRACT

A complex finite element analysis can produce large amount of data that is problematic to post-process in reasonable time. This paper describes application of Singular Value Decomposition (SVD) to the compression of results from finite element solvers. Although the idea of image compression method is an inspiration for this research work, the SVD compression algorithm used for compression of images cannot be directly used for FEM results. Differences and implementation challenges are discussed in the text. Quality of approximation is more important in scientific field than in computer graphics where the most significant factor is the human perception of the resulting image. Error estimation methods used during compression of finite element results are presented. The focus is also on the algorithm performance. SVD is a very computational intensive method. Therefore, various optimization techniques were investigated, e.g. randomized SVD. The method leads to the lower memory consumption, 10% of the original size or less, with negligible compression error.

1. Introduction

Amount of data produced by a complex finite element analysis can be enormous and typical personal computer is not capable to store, process and visualize the results in reasonable time. Singular Value Decomposition is a well known factorization method that provides rich information about matrix systems. One of its many applications is the image compression where it can significantly reduce the size of the data representing an image while preserving the quality of image appearance.

The effort to reduce size of resulting data from complex finite element analyses to accelerate (or even make possible) post-processing using common personal computer is not new. In [1], there is one possible direction presented for research in this area – the replacement of discrete data produced by finite element solver by continuous functions. These approximation functions can be then described by a small number of parameters. The main goal is to find the areas in domain where the output discrete function has predictable development and can be easily replaced by a simple continuous function, e.g. linear function. Although this approach has remarkable results for some classes of functions it is not applicable to a general problem. For some special cases, such as functions with discontinuities, the method has poor results because approximation error is too high and – what is more important – it cannot be determined in advance.

In this study it was decided to apply purely algebraic approach. Various methods were considered, e.g. discrete wavelet transform [2] and discrete cosine transform [3], for their successful use in image compression. SVD method [4–6] was chosen as the foundation of the compression algorithm. Considering that the results from the FEM analyses can be viewed as a series of arbitrary rectangular matrices, the implementation of compression algorithm based on SVD is straightforward as it can be applied to any rectangular matrix.

Compression methods usually yield approximated data. In the following text, the term *approximation error* denotes an error resulted from compression, i.e. difference between original results of FEM analysis and their compressed form. It should not be confused with the error of the Finite Element Method itself that yields approximate solution to mathematical problems used to model physical reality. To quantify the approximation error, several error metrics were investigated. In [7], there are some of them used in similar area of research. The ability to control the quality of compression was a key requirement for the implementation of the compression algorithm.

The quality of any compression method depends on the nature of input data. Therefore, several different non-trivial finite element analyses should be used as benchmarks for an implementation of the compression algorithm. Also, the size of input data (the output from the FEM analysis) should be large enough to test the algorithmic complexity under heavy load. Details about the analyses that were used as benchmarks can be found in [8–11].

Performance is very important aspect for development of the

* Corresponding author.
E-mail addresses: stepan.benes@fsv.cvut.cz (Š. Beneš), jaroslav.kruis@fsv.cvut.cz (J. Kruis).

https://doi.org/10.1016/j.advengsoft.2017.12.007
Received 28 June 2017; Received in revised form 30 November 2017; Accepted 28 December 2017

compression algorithm. SVD being very computational intensive, randomized algorithms for SVD decomposition were investigated [12–15]. Especially, the implementation described in [16] was used in the compression algorithm for the optimization. Memory consumption and the suitable format of the output data from the compression algorithm also influence overall usability of the resulting work. Efficient data storage is connected with data structure [17,18]. This area will be subject of further research.

Although the idea to use SVD for data compression is not new, it is applied in an entirely new area of post-processing of results from the FEM analysis. In contrast with image compression, where SVD was used in a lossy compression algorithms, the area of post-processing puts great emphasis on the mathematical accuracy of the approximation instead of the human perceptions of visualizations. Also, the storage and dimensions of the input data, and the variance of the values in data, are very different. The main contribution of this paper is therefore to explore this area and to offer a description of an implementation of the algorithm that is successfully used in the post-processing of large data.

The rest of the paper is organized as follows. Section 2 contains mathematical background of the SVD compression (i.e. SVD method itself, its use in low-rank matrix approximation, and description of the randomized SVD method). Implementation of the compression algorithm is presented in Section 3. Section 4 summarizes the results of the benchmarks that were designed to measure quality of the output from the compression algorithm, and also the performance of its implementation. The paper is concluded in Section 5.

2. Mathematical background

Singular value decomposition is based on a theorem from linear algebra which says that a rectangular matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ can be decomposed into the product of three matrices - an orthogonal matrix $\mathbf{U} \in \mathbb{R}^{m\times m}$, a diagonal matrix $\mathbf{S} \in \mathbb{R}^{m\times n}$, and the transpose of an orthogonal matrix $\mathbf{V} \in \mathbb{R}^{n\times n}$:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}, \tag{1}$$

where $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$, $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$. The columns of $\mathbf{U}$ are orthonormal eigenvectors of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, which are called the left singular vectors. The columns of $\mathbf{V}$ are orthonormal eigenvectors of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ called the right singular vectors. $\mathbf{S}$ (sometimes referred to as $\mathbf{\Sigma}$) is a diagonal matrix containing singular values in descending order, which are at the same time the nonzero square roots of the eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

SVD can be seen as a method for transforming correlated variables into a set of uncorrelated ones. At the same time, SVD is a method for ordering the dimensions based on variation and identifying the dimension with the largest variation. Once this dimension is identified, it is possible to find the best approximation of the original data points using fewer dimensions. Hence, SVD can be seen as a method for data reduction/compression.

2.1. SVD compression

This is the basic idea behind SVD: taking a high dimensional, highly variable set of data points and reducing it to a lower dimensional space that exposes the substructure of the original data more clearly and orders it from the largest variation to the least. What makes SVD practical for data compression applications is that variation below a particular threshold can be simply ignored to massively reduce data with assurance that the main relationships of interest have been preserved.

The objective of a compression algorithm is to reduce amount of data representing FEM results and also the ability to reconstruct original data from its smaller representation. This saves storage capacity and also accelerates the data transfer between computers as the analysis itself and the post-processing of results is sometimes done on different workstations.

A compression method can be lossy or lossless. Lossless methods are able to fully reconstruct original data. Lossy methods, on the other hand, produce only approximations of original data.

SVD is used in this paper as a part of the compression algorithm. The SVD method applied to arbitrary matrix produces decomposition that consists of corresponding singular values and singular vectors. This process is fully reversible (with the assumption that the numerical errors are negligible). The original matrix can be reconstructed by the multiplication of decomposed parts. However, the compression algorithm is based on modification of decomposition to create low-rank approximation matrix. The reconstructed matrix slightly differs from the original matrix and algorithm therefore performs lossy compression.

2.2. Low-rank approximation matrix

From the definition of SVD in (1) and from the properties of SVD, the fact follows that a matrix can be represented in the form of its SVD components as a sum of k rank-1 matrices

$$\mathbf{A} = \sum_{i=1}^{k} s_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}, \tag{2}$$

where s_i is the ith singular value of matrix $\mathbf{A}$, $\mathbf{u}_i$ and $\mathbf{v}_i$ are corresponding singular vectors of matrix $\mathbf{A}$, and $k = \min(m, n)$. Considering the fact that singular values are ordered $s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_k$, the above formula implies that the first term of the sum would have the highest contribution and the last term would have the lowest contribution to matrix $\mathbf{A}$. Therefore, if we take only first r members of the above summation we get an approximation matrix

$$\mathbf{A}' = \sum_{i=1}^{r} s_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}. \tag{3}$$

Quality of approximation depends on the magnitude of the singular values omitted from the approximation formula, namely $s_{r+1} \ldots s_k$. The compression algorithm is based on an assumption that the first singular value is order-of-magnitude higher than singular values at the end of the decomposition sequence. In special cases, when $r = k$, or $s_i = 0$ for all $i > r$, the omitted singular values do not contribute to the sum and the compression is therefore lossless. In other cases, approximation error has to be calculated and taken into account to avoid loss of important details in data.

The main goal of the compression algorithm is to find a compromise between low approximation error and high compression ratio c which is calculated using the formula

$$c = \frac{r(m + n + 1)}{mn}, \tag{4}$$

where m is the number of rows and n is the number of columns of matrix $\mathbf{A}$. Explanation of the compression ratio formula is best done using Fig. 1. Light color represents the part of matrix decomposition that is to be stored in the output file as a low-rank approximation of the input.

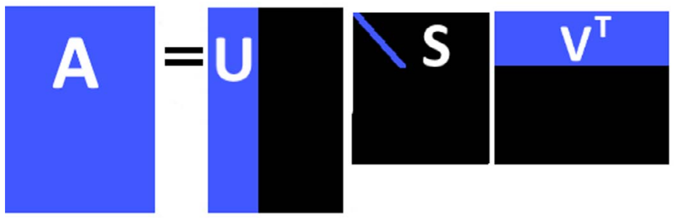

Fig. 1. Decomposition of input matrix **A** into diagonal matrix of singular values **S** and matrices of left and right singular vectors. Light color illustrates low-rank approximation.