Parallelization of computational algorithms for optimization problems with high time consumption

Milan Vaško[1,*], *Marián* Handrik[1], *Alžbeta* Sapietová[1], *Jana* Handriková[2]

[1]University of Žilina, Faculty of Mechanical Engineering, Department of Applied Mechanics, Univerzitná 8215/1, Žilina, Slovak Republic

[2]Armed Forces Academy of General M. R. Štefánik, Department of Informatics, Liptovský Mikuláš, Slovak Republic

Abstract. The paper presents an analysis of the use of optimization algorithms in parallel solutions and distributed computing systems. The primary goal is to use evolutionary algorithms and their implementation into parallel calculations. Parallelization of computational algorithms is suitable for the following cases - computational models with a large number of design variables or cases where the objective function evaluation is time consuming (FE analysis). As the software platform for application of distributed optimization algorithms is using MATLAB and BOINC software package.

Keywords: ADINA, MATLAB, BOINC, optimization algorithms, parallelization, distributed computing systems, evolutionary algorithms, FE analysis

1 Introduction

Solving of the optimization problem is generally based on the repeated evaluation of the objective function. Evaluation of the objective function represents FE analysis of the issue in the optimization problems in continuum mechanics [1, 2]. It is necessary to implement several types of analyses based on FEM for complex optimization problems solving (stress analysis to determine the state of stress distribution, modal–frequency analysis to determine the natural frequencies, loss of stability analysis, etc.) [3, 4].

To obtain the optimal solution is required tens to hundreds of thousands of objective function evaluations, depending on the used optimization method and the number of design variables [5]. The solution of such problems is unrealistic using conventional optimization algorithms because this solution could take several months or years.

For the purpose of more efficiently solve the optimization problem it is necessary to modify optimization algorithms. The ADINA software package will be used for evaluating the objective function value in the implementation of parallel optimization algorithms [6, 7]. It can be defined parameters and create a parametric model in the input file. This enables easy change of design variables in optimization process.

* Corresponding author: milan.vasko@fstroj.uniza.sk

Reviewers: *Radim* Halama, *Ján* Vavro *Jr.*

2 Computational systems for optimization problem solution

Using the parallel solution of linear equations in the FEM analysis is a fundamental possibility of increasing the speed of the optimization problem solution [1]. The two fundamental architectures of computational systems can be used in this case:

- Symmetric Multi Processing (SMP) technology – it means multiprocessor and multicore computers,
- Distributed Multi Processing (DMP) technology – this is a group of computers that are connected to a computer network and computers work together to solve problems.

Iterative solvers are used for solving systems of linear equations in large-scale problems. Parallel implementation of iterative solvers is effective only for a relatively small number of compute cores that are used for the problem solution. Increasing of the computing power is sometimes possible by using graphic accelerators. In the case of more required and solving tasks and provided adequate computing resources power to implement they the following calculation procedure can be used:

- Serial problem solving using the whole computing power – in this case the nodes are often only partially used.
- Parallel solutions of multiple tasks on the smaller number of computational cores – individual computing nodes are used more efficiently.

Practical tests have shown that in case of sufficient capacity of computing resources the parallel solution in terms of duration of solving across a series of tasks appears more efficient. Alternatively, increasing the speed of solving the optimization problem is the use of the above properties and modification of the optimization algorithms to the parallel implementation.

3 The optimization algorithms

Optimization methods can be divided into analytical and iterative. If in solving complex optimization problems of continuum mechanics to evaluate the objective function is using commercial FEM program, it is possible the iterative methods use only in the optimizing process. Iterative optimization algorithms can be divided into three basic groups [2]:

- the direct methods,
- the first order methods,
- the second order methods.

The group of direct methods can be divided into two subgroups – methods of random search and evolutionary algorithms. Their parallelization is simple – it is sufficient to implement a parallel evaluation of objective function, i.e. multiple FEM analyses solution work simultaneously. A subset of deterministic methods after their parallelization is transformed into the first subset or the gradient methods. Parallelization of the first order methods is based on the process of numerical differentiation parallelization [7, 8]. Based on the possibility of parallelization the iterative optimization algorithms can be divided into the following groups.

Algorithms that cannot be parallelized - this group can include algorithms where parallelization is possible only in a few steps of the algorithm (Simplex algorithm, in which is only possible to parallelize the compilation of the initial simplex).

Algorithms with the interaction - this group consists of algorithms in which needs to be completed one step of algorithm to algorithm can continued in the calculation (gradient methods). With these methods it is possible to parallelize the gradient or Hess matrix

compilation while algorithm must wait for their establishment. Therefore, limiting factor for running the program is waiting for compilation the last element of the gradient or Hess matrix. Suitable computational resources for this type of algorithms are those where each computing node has the same computing power. Then, the powerful computational nodes do not have to wait for less powerful computing nodes.

Algorithms without the interaction - algorithms included in this group have to solve a high degree of tolerance to time and return the objective function solution (evolutionary algorithms). The descendants group with appropriate genetic variations is generating in the vicinity of the parent. The value of the objective function will be subsequently evaluated for this group of descendants. Member with the best objective function value becomes the new parent.

The following modification can be used for parallelization of evolutionary algorithms: Packet of descendants is created into which shall be supplemented continuously descendants generating on the base of current parents. These descendants are waiting to evaluate the objective function. The next step is parallel evaluation of descendants waiting in the packet and comparing the value of the objective function with a parent. If a descendant has a better objective function value than the current parent this child becomes the new parent.

In the packet of descendants may be found simultaneously descendants from the different parent generations but better parent can be found in the previous generation as is the current parent. Parallel evaluation of the descendant's objective function is mutually independent and sufficient condition the reasonable length of time for return the objective function. This condition allows into simultaneous evaluation include the computing resources with significantly different computing power (Fig. 1).

4 Grid computing using evolutionary algorithms

Gradient and evolutionary algorithms are significant in terms of optimization algorithms parallelization with the use of a commercial FEM program for evaluating the value of objective function. Problems of continuum mechanics cannot be considered as task with the one global extreme. More local extremes are often found in the solution area.

The term *grid computing* represents group of different computers that work together to solve complex computing tasks. Complex task is to find the optimal solution of the problem in terms of the optimization process. Partial task is to determine the value of the objective function for the vector of design variables. In evolutionary algorithms it represents determination the value of objective function for a specific descendant. There are a number of software platforms for the implementation of grid computing today.

BOINC (Berkeley Open Infrastructure for Network Computing) has been used to implement a distributed computing model. It's open source system for distributed computing. The robustness, flexibility and universality of the system is verified on a number of world famous projects in the area of grid computing (SETI, Milkyway, Einstein, etc.) [9].

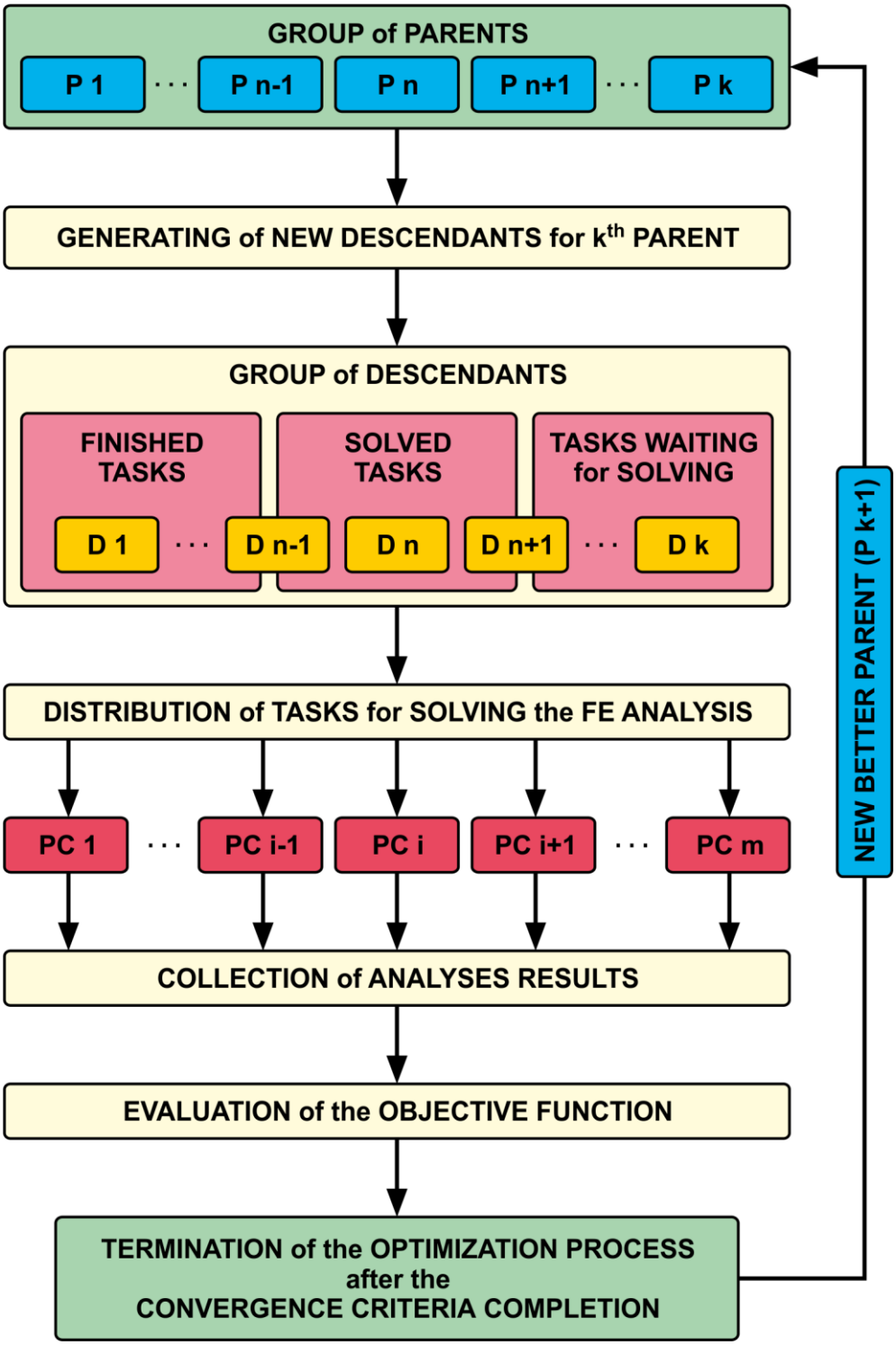

Fig. 1. Scheme of the optimization process for the grid infrastructure

Computing system consists of server and client part. The server part is designed to ensure the following functions:

- generate work units (work unit generator),
- insertion of work units into a set of tasks to send (feeder),
- identification of client's request and schedule its compliance (scheduler),
- client requirements completion – download work units and upload results (CGI scripts in Apache),
- comparison of the results for individual work units (validator),
- processing results after their validation (assimilator),
- provision of communication and reporting information about the process of the work units solution (transitioner),
- release disk space by deleting of solved work units (file deleter).

The client part is designed to ensure the following functions:

- download the main program for work unit solving,
- monitoring updates and ensure downloading a newer version,
- enable logging to the project or multiple projects,
- allocation of free system resources for their use,
- sending requests for the allocation of work units and their download on the client computer,
- send the information about status and results of solution.

A computer running Debian Linux 6.0 has been used to implement a distributed computing system. BOINC was installed directly into the system without using virtualization. The following changes were necessary perform to adjust the BOINC system for needs of optimization model implementation:

- change the general system settings,
- adjust the work units generator,
- create a client application,
- adjust the processing of the results, modifying the assimilator.

General settings change consists of changing the approach of the feeder to the prepared work units. The strategy of work units' choices was changed for insertion in the system FIFO (First In First Out). Work units, i.e. individual descendants are inserting from the oldest to newest for solving in the evolutionary strategy.

At the editing of work units generator it is based on the assumption that its task is generate new descendants with a random genetic mutation in the best parent's neighborhood. The generator monitors the number of prepared work units waiting to be processed. The best parent parameters are loaded and runs generate a new group of descendants if the number of prepared work units decreases under the specific bound.

It is monitoring before generating of descendants if in terms of the solution convergence the generating is needed. The creation of work unit consists of generating random genetic mutations, preparation of input files for the ADINA preprocessor, preparing files for the ADINA postprocesing and compression of all files to the "7zip" archive.

The advantage is that at any modification in the optimization task there is always necessary to transfer only one file. The optimization problem change is therefore realized by the change of the input files for work unit's generator on the server side. Therefore is not necessary to change the work unit's generator. The role of the client application is to ensure the following functions:

- extract the files from the work unit archive,
- run ADINA preprocessing and then solving tasks in ADINA,
- write selected results into files by running ADINA postprocesing,

- archiving initial and executive programs by “7zip” into a single file for transmission to the server.

The assimilator modification is in preparing them to execution the following steps:

- The result of the analysis solved by FEM in the program ADINA can contain multiple numeric data recorded in different files. The assimilator compiles objective function value from the partial results and then determines if the evaluated descendant becomes new parent.
- If it becomes the new parent then into the relevant files are written new parent parameters and its objective function value.
- In the evaluation of the each work unit convergence criterion is checked and in the case of its compliance gives the instruction to the termination of generating work units.
- The condition was used in the choice criteria of convergence if enough new descendants were generated without finding a new better parent.

It is necessary to make appropriate adjustments in the assimilator when is changing the form of the target function.

5 Optimization process and FE analyses

The optimization algorithm programmed in MATLAB using the BOINC system and evolutionary optimization algorithms was developed to test the proposed methodology [9-11]. The algorithm was tested on the problem of optimizing the wall thicknesses of the shell structure. The tests were performed on a simple structure consisting of a beam with an "I" cross section. The beam was loaded with continuous (distributed) load and supported at the ends.

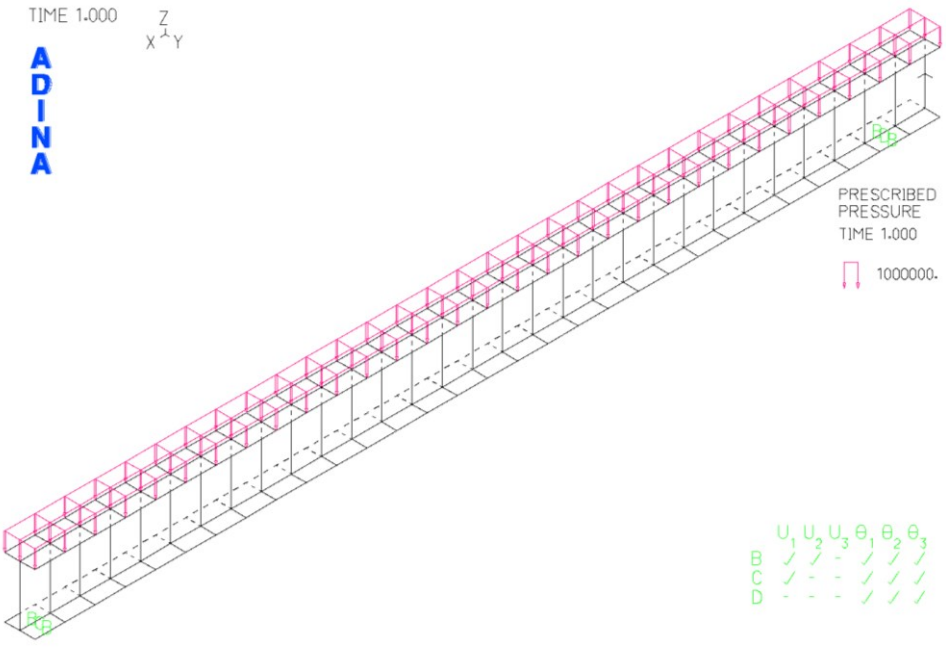

Fig. 2. Geometric model with boundary conditions

The geometric model of the beam was divided into 30 parts and each part consisted of three independent thicknesses of the beam wall. Thus, the total number of optimization variables was ninety.

An elastic material model for steel was used in the calculation. The starting thicknesses of the shell structure were 0.01 m. Created geometric model with boundary conditions is shown in Fig. 2. Tetragonal mesh of shell elements with the element size of 0.002 m was used in the finite element model. The total number of elements was 76 500 and the total number of nodes was 77 353. The target function has been choosing so that the maximum value of equivalent stresses of 100 MPa in each part of the structure was reached at optimum. A secondary condition was that the minimum thickness of the shell structure was not less than 0.001 m.

The distribution of von Mises equivalent stresses before the optimization process is shown in Fig. 3. The maximum value of the stress was 313 MPa and the minimum stress was 0.2 MPa.

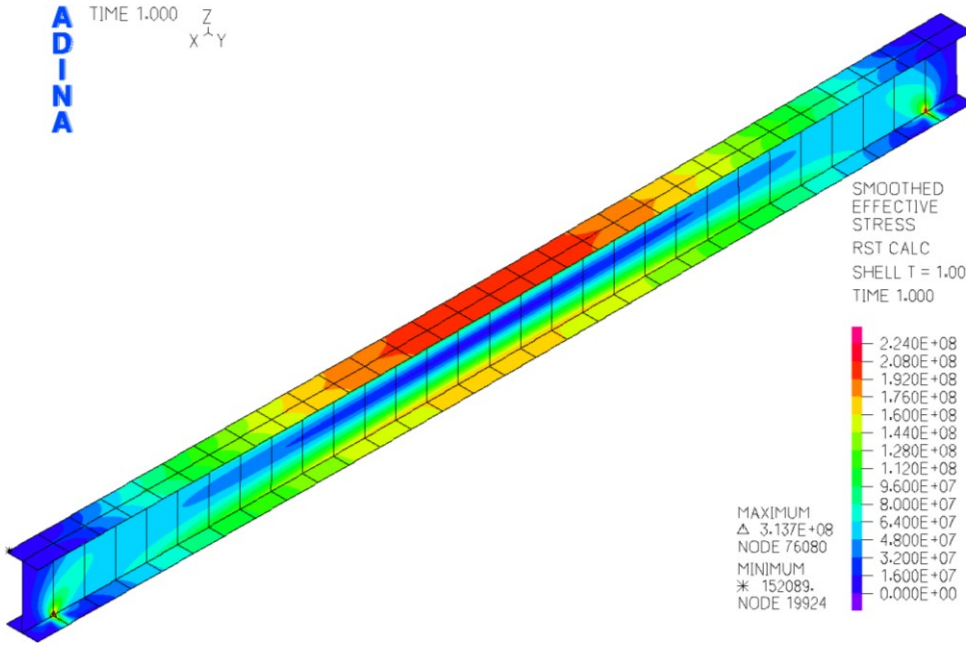

Fig. 3. Distribution of von Mises equivalent stresses before the optimization process

The resulting distribution of von Mises stress after the optimization process is shown in Fig. 4. The maximum stress reached a value of 100 MPa and minimum stress was 0.02 MPa in this case. The resulting wall thickness of the individual parts of the beam after optimization is shown in Fig. 5. The maximum wall thickness reached 0.2988 m and the minimum wall thickness was 0.001 m.

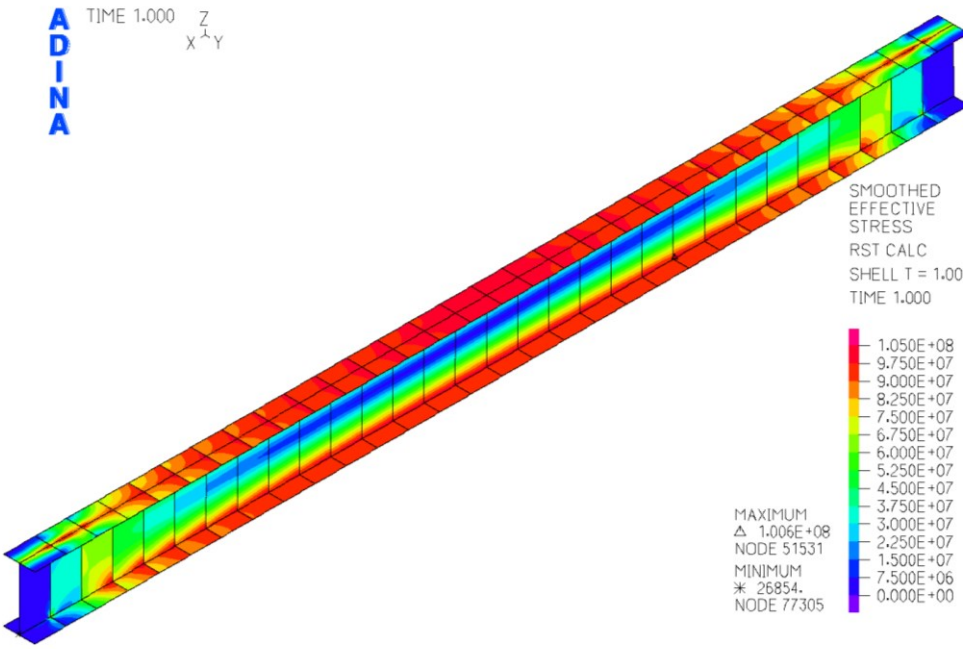

Fig. 4. Resulting distribution of von Mises stresses after the optimization process

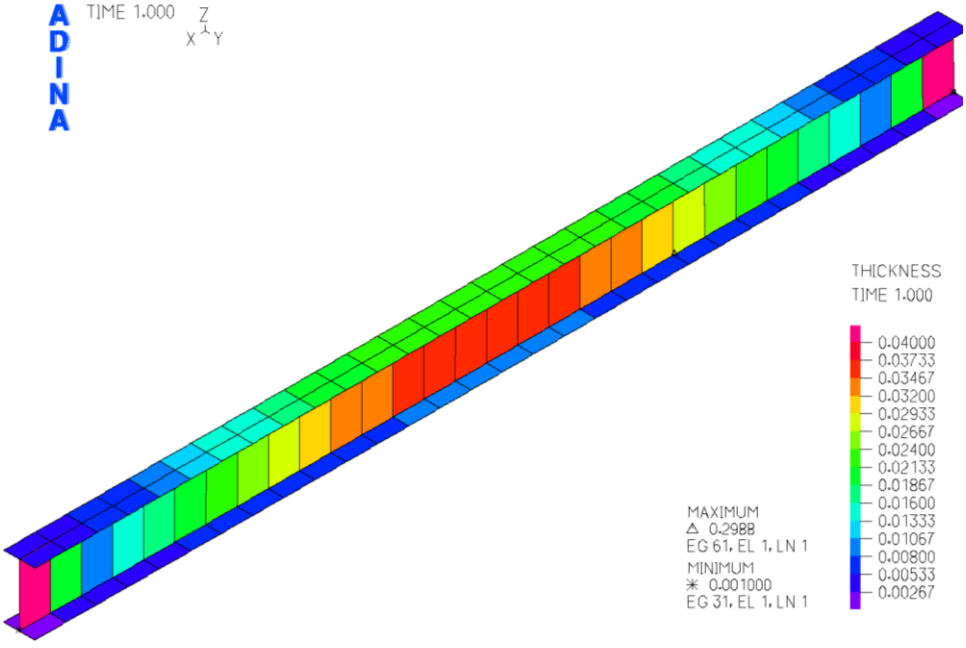

Fig. 5. Resulting wall thickness of the individual parts of the beam after optimization

Low stress values after optimization were obtained in the end segments of the beam but in places of boundary conditions the maximum stress value was reached [12]. The optimization process was conducted on a group of 26 computers. The calculation lasted 108 hours and the 69 542 iterations of the optimization task was performed.

Conclusions

Use of the BOINC in combination with MATLAB as a control system for grid computing allows very fast response of the system at the appropriate setting of its parameters (i.e. the short time between the end of the FEM analysis and processing of the results in the assimilator for the work units generator). This allows the implementation of appropriate and adapted evolutionary algorithms that require a higher level of feedback between the generated descendants and actual parents.

A small claims rate on system resources, parallel realization of the work units generation and parallel processing of the results enables involving the order of several thousand to tens of thousands of PCs in FEM analyses necessary for solving the optimization problem. Robust computing infrastructure allows the realization of optimization problems for finding the global extreme in continuum mechanics problems with hundreds of design variables in relatively short time.

This work is supported by VEGA 1/0795/16 “Development of effective methods for correction and optimization of coupled mechanical system”.

References

1. K. J. Bathe, *Finite Element Procedures*. (Prentice Hall, New Jersey, 1982)
2. M. Sága, M. Vaško, N. Čuboňová, W. Piekarska, *Optimisation Algorithms in Mechanical Engineering Applications*. (Pearson Education Limited, 291 p., 2016)
3. L. Jakubovičová, M. Sága, *Computational analysis of contact stress distribution in the case of mutual stewing of roller bearing rings*. Novel Trends in Production Devices and Systems, Applied Mechanics and Materials **474**, 363-368 (2014)
4. A. Vaško, L. Hurtalová, M. Uhríčik, E. Tillová, *Fatigue of nodular cast iron at high frequency loading*. Materialwissenschaft und Werkstofftechnik **47** (5-6), 436-443 (2016)
5. P. Oršanský, B. Ftorek, P. Ďurčanský, *Mathematical model of a closed hot air engine cycle using MATLAB Simulink*. XIX. The Application of Experimental and Numerical Methods in Fluid Mechanics and Energetic 2014, AIP Conference Proceedings **1608**, 173-176 (2014)
6. ADINA, *Theory and Modelling Guide*. [help manual]. (Watertown, 2010)
7. P. Pecháč, M. Sága, *Memetic algorithm with normalized RBF ANN for approximation of objective function and sedondary RBF ANN for error mapping*. Procedia Engineering **177**, 540-547 (2017)
8. M. Vaško, M. Sága, M. Handrik, *Comparison Study of the Computational Methods for Eigenvalues IFE Analysis*. Appl. and Computational Mechanics **2** (1), 157-166 (2008)
9. *Berkeley Open Infrastructure for Network Computing* (*BOINC*). [online] http://boinc.berkeley.edu/
10. *MATLAB*. [online] https://www.mathworks.com/products/matlab.html

11. L. Žuľová, R. Grega, J. Krajňák, G. Fedorko, V. Molnár, *Optimization of noisiness of mechanical system by using a pneumatic tuner during a failure of piston machine*. Engineering Failure Analysis **79**, 845-851 (2017)
12. A. Sapietová, V. Dekýš, M. Sapieta, P. Pecháč, *Application of computational and design approaches to improve carrier stability*. Procedia Engineering **96**, 410-418 (2014)