Efficiency of Abaqus2Matlab toolbox for structural optimization problems

V T Pham¹, Q V Vu¹*, G Papazafeiropoulos²*, V T Ngo³

¹Faculty of Civil Engineering, Vietnam Maritime University, 484 Lach Tray Street, Haiphong city, Vietnam.
²Dept. of Structural Engineering, National Technical University of Athens, Zografou, Athens 15780, Greece.
³Faculty of Civil Engineering, Thuyloi University, 175 Tay Son Street, Dong Da District, Hanoi city - 100000, Vietnam.

E-mail: vietvq@vimaru.edu.vn; gpapazafeiropoulos@yahoo.gr

Abstract. In recent years, optimal structural design is one of the most interesting fields in the engineering science. With the development of the computational tools, many optimization methods have been proposed. This paper introduces an efficient toolbox namely Abaqus2Matlab which can be used for any structural optimization problem. This computational tool allows for automatic linking between Abaqus and Matlab, in terms of transferring data from Abaqus to Matlab and vice versa, and creating the necessary files for the optimization procedure. In the optimization problem, Abaqus plays the role of the solver, while Matlab plays the role of the optimizer. In order to demonstrate the efficiency of Abaqus2Matlab toolbox, an optimization procedure is employed to find the optimum sizing of the member cross sectional areas for the benchmark 10-Bar truss and 25-Bar truss problems. The results obtained from this method are validated with various studies from the literature.

1. Introduction

Structural optimization has great potential for the construction industry since it is used to design material-efficient or cost-effective structures. When applying structural design optimization, not only the construction cost, but also the engineering cost is reduced, by automating the repetitive task of sizing structural members. In addition, structural optimization can lead to innovative design solutions for specific structural materials or components. There are three structural optimization strategies consisting of size, shape, and topology optimization. In order to solve these structural optimization problems, many structural optimization algorithms have been proposed. For steel, concrete and steel-concrete composite structures, much research related to its structural optimization has been implemented in the literature with various efficient algorithms and techniques. Some of the well-known optimization algorithms are ant colony optimization (ACO) [1], genetic algorithm (GA) [2], harmony search (HS) [3], particle swarm optimization (PSO) [4], simulated annealing (SA) [5], etc. Although there has been remarkable enhancement in the study of these optimization algorithms, most of their applications are limited to linear structures where nonlinear inelastic behavior of the structure has not been taken into consideration. This may lead to unrealistic and non-economic optimum designs. In order to consider the nonlinear inelastic behavior of steel structures in structural optimization, Truong et al. [6], Truong and Kim [7-8] proposed efficient methods where advanced
analysis using PAAP (which is introduced by Kim and Chen [9]) was used to analyze steel frames and trusses considering their nonlinear inelastic behavior, and then the results taken from PAAP were employed for optimization procedure using some improved optimization algorithms such as micro-GA and improved DE. The optimum results obtained from these proposed methods were demonstrated to be efficient and realistic. However, the computational tool PAAP is not open source and is difficult to use; therefore, the application of these methods in practical design still remains a challenge.

This paper aims to investigate the efficiency of Abaqus2Matlab tool which is developed by Papazafeiropoulos et al. [10] in structural optimization. This toolbox is powerful since Abaqus2Matlab allows to run Abaqus remotely from Matlab and to post-process the results, thus providing a link between the two well-known packages Abaqus and Matlab. Abaqus plays the role of solver which easily analyzes any structure considering its nonlinear inelastic behavior, while Matlab plays the role of optimizer which can efficiently apply many optimization algorithms.

2. Structural optimization with Abaqus2Matlab

Structural optimization is a research field dealing with optimal design of load-carrying mechanical structures. The standard form of a structural optimization problem is as follows:

Minimize:
\[ f(x) \]
Subject to:
\[ g_i(x) \leq 0, \quad i = 1, \ldots, m \]
\[ h_j(x) = 0, \quad j = 1, \ldots, p \]

Abaqus2Matlab provides an interface between Matlab which serves as the optimization environment, and Abaqus, which serves as the structural analysis solver. In such cases where interfaces are used for coupling the optimization software and the solver software, it is of paramount importance that the user takes into consideration two major components of the optimization procedure which are formulation and way of solving the optimization problem and (2) sensitivity and approximation issues. These are described below, along with various recommendations to the Abaqus2Matlab user, in order to enjoy the best possible solutions to optimization problems using the Abaqus2Matlab software.

2.1. Formulation and solution of the optimization problem

The formulation of an optimization problem affects the success of the optimum design process in terms of computational effort and quality of results. Numerical difficulties result mainly from the following reasons:

• Usage of highly nonlinear objective or constraint functions. The gains in the convergence rate are apparent when linear formulations inside the objective and constraint functions of the optimization problem are used. In some cases, nonlinear formulations can be converted into equivalent linear and thus simplify the problem.

• Large differences between the magnitudes of the design variables, objective function(s) and constraint function(s). This problem can worsen if the software involved in the optimization procedure are not numerically robust. The best option in this case is to normalize the design variables, objective function(s) and equality constraint(s) to order 1 by scaling, and to normalize the inequality constraint(s) by the maximum or minimum value used to form them.

• Determination of the set of active constraints. If all the constraints are considered during the search process, the computational effort may be very high, whereas the consideration of only the constraints that are active or nearly active at a trial solution may lead to spurious oscillations and therefore inability for convergence. An appropriate and robust methodology for the consideration of the active set of constraints is a vital component of an optimization algorithm and must be carefully selected.
• The Abaqus structural analysis model depends on the formulation of the optimization problem running in Matlab, in which it participates. While sometimes a detailed Abaqus model is required to verify a case, if the same Abaqus model is involved in an optimization procedure in Matlab, it needs to be appropriately simplified and/or reduced, so that the increase in the computational load is not prohibitively high, given the fact that a large number of Abaqus analyses are induced by the optimization algorithm running in Matlab.

• The formulation of the optimization problem in Matlab sometimes depends on the Abaqus model. If the sensitivity of the Abaqus FE analysis results with respect to the design variables is low, then there is room for the use of a relatively simple and more straightforward optimization algorithm in Matlab; otherwise, depending on the complexity of the Abaqus model, the optimization algorithm that will be used in Matlab must meet the Abaqus model requirements.

• The Abaqus FE analysis and Matlab optimization procedure can be integrated on an step-by-step basis, especially when the Abaqus analysis is highly nonlinear. It is possible to combine the FE analysis and optimum design iterations in a single iterative process using Abaqus2Matlab.

2.2. Sensitivity

This aspect is related either to high gradients of the objective and/or constraint function(s) with respect to the design variables, or to the existence of jumps in the variation of these functions. The sensitivity of an optimization algorithm is influenced by the efficient calculation of derivatives of the objective and constraint function(s), with respect to the design variables. The importance of these derivatives is apparent, as they are usually used for

• Approximate constraint evaluations to reduce the computational effort associated with exact evaluations
• Evaluation of the direction at which the optimization algorithm will proceed in each step to reach a solution which is better than the current

The sensitivity of the derivatives clearly affects the accuracy of the optimum solution as well as the stability of the optimization algorithm. In cases that the sensitivity of the derivatives is high, the following options are proposed:

• Suitable re-formulation of the optimization problem so that the optimization domain contains fewer or no irregular (singular) points. Quantitatively, estimation of the degree of irregularity of the search region is a matter of experience and can be crudely calculated by the ratio of the largest to the smallest eigenvalue of the Hessian matrix of the objective function at the optimal point, only in the case of unconstrained optimization problems.

• Substitute of the used optimization algorithm with another algorithm that uses fewer derivatives and/or has superlinear convergence rate, unless the computational load per iteration becomes prohibitively high. If a sequence $x_1, x_2, ..., x_n$ converges to a value $r$ and if there exist real numbers $\lambda > 0$ and $\alpha \geq 1$ such that

$$
\lim_{n \to \infty} \frac{|x_{n+1} - r|}{|x_n - r|^\alpha} = \lambda
$$

then $\alpha$ is the rate of convergence of the sequence.

From the above it is apparent that the selection of the optimization algorithm is case-dependent, and it is nearly impossible to assert without ambiguity that an algorithm is generally better than another.

3. Structural analysis solver function using Abaqus2Matlab

The main purpose of Abaqus2Matlab as an interface within an optimization procedure is to play the role of the structural analysis solver in an automated and reliable way, non-prone to errors. For this purpose, Abaqus2Matlab follows a certain workflow, which is different among its various
components, e.g. it is different between Abaqus2Matlab/fil2Matlab (responsible for processing Abaqus results (*.fil) files), and Abaqus2Matlab/odb2Matlab (responsible for processing Abaqus output data base (*.odb) files). In this study, the fil2Matlab component is used. The workflow is shown in Figure 1 and is executed as follows:

• The Abaqus input file that corresponds to the current values of the design variables $x$ is generated. This job is done by the Abaqus2Matlab function InpFileConstr.m. This function apparently depends on the Abaqus model and the way that it is parameterized. $x$ is a vector containing the current values of the design variables. The function InpFileConstr.m does not give any output. The following option with its parameter has to be specified in the Abaqus input file in order for the results (*.fil) file to be generated:

... *FILE FORMAT, ASCII ...

• After the Abaqus input file is generated, it is run remotely by Abaqus2Matlab, through the function runAbaqusAnalysis.m. This function accepts the name of the Abaqus input file to be run, an upper time limit $t_{ub}$ and a lower time limit $t_{lb}$. Abaqus execution starts and afterwards Matlab execution is halted, waiting until any of the following events happens:

(1) Abaqus execution starts normally and the Abaqus lock (*.lck) file is deleted. This is the normal case.

(2) Abaqus execution starts normally and $t_{ub}$ is exceeded (for some reason Abaqus execution is lagged). In this case, Abaqus analysis is automatically terminated and executed again.

(3) Abaqus execution starts but the Abaqus lock (*.lck) file is not generated, and $t_{lb}$ is exceeded (the execution of the process SMASimUtility.exe is lagged). In this case, the process SMASimUtility.exe is automatically killed and the Abaqus analysis is executed again. In this function Java variants of various Matlab commands are used where possible, in order to avoid memory leaks which may cause lags or crashes of Abaqus execution. Furthermore, the Java commands are proven to be much more accurate than the corresponding Matlab commands.

• After the Abaqus analysis has been completed, control is passed back to Matlab and then the Fil2str.m function is used to read the information contained in the Abaqus results (*.fil) file in ASCII format and assemble it into a one-row string. The function Fil2str.m initially opens the results file with reading only permission, then reads the data included in this file as an assembly of strings within a cell structure, and finally it removes any newline and carriage return characters from these strings, eventually resulting in a single-row string that is passed as output (output argument $s$). Provisions are taken in order to ensure that the Fil2str.m function works for any Matlab version that the user may be running.
The single line string that is output by the function Fil2str.m is processed by another suitable function that is specified by the user, depending on the type of the desired Abaqus results. Of course, the string may contain more than one types of data (such as nodal displacements and element stresses for example), for the extraction of which different functions must be selected by the user then using the open source version of Abaqus2Matlab (v.1.0). This feature has been improved in the second version of Abaqus2Matlab, where a single function is used for any type of Abaqus results that are extracted, and the user has only to specify the record key as an input argument to this function. In the first (open source) version of Abaqus2Matlab there is a standard naming convention of these functions, i.e. RecXXX.m as shown in Figure 1, where XXX is replaced with the record key corresponding to the desired result. For example, in this study the function Rec101.m was used to extract nodal displacement results (record key 101), whereas the function Rec13.m was used to extract element section forces (record key 13).

4. Use of Abaqus2Matlab for optimum structural design

The aforementioned description is applicable for the implementation of any structural analysis solver function using Abaqus2Matlab. Consequently Abaqus2Matlab can be used for any structural optimization problem that is solvable in Abaqus. In the following the implementation of Abaqus2Matlab for the solution of two truss optimization problems will be shown. The Matlab fmincon function was used as the optimizer for the solution of the two truss optimization problems considered in this study.

There are various patterns for using the Abaqus2Matlab solver within the framework of the Matlab fmincon function. Figure 2 presents the relevant schematic flow diagram of possible ways of inclusion of the Abaqus2Matlab solver. It is possible to use Abaqus2Matlab for the objective function evaluation (Case 1) or not (Case 2). The same happens with the constraint function evaluation. Therefore the Abaqus2Matlab solver that is presented in Figure 1 can be used as shown in Figure 2 for...
the evaluation of either the objective function, or the constraint function, or both of them. Since we are dealing with a structural optimization problem, it is necessary that at least one of the two functions (objective and constraint functions) implements the flowchart of Case 1. In the present study, the Abaqus2Matlab was used only in the constraint function evaluation, i.e. Case 2 and Case 1 were used for the objective and constraint functions respectively.

Figure 2. Flowchart of the possible uses of Abaqus2Matlab for the objective and constraint function evaluations within the framework of the Matlab fmincon function.

Instructions about the proposed methodology in order to setup and solve a structural optimization problem are shown in Figure 3.
Figure 3. Suggested methodology for setting up and performing a structural optimization problem using Abaqus2Matlab.

5. Examples

In order to demonstrate the efficiency of optimization procedure using Abaqus2Matlab presented above, this section shows a comparison of the optimum results obtained from optimization procedure using Abaqus2Matlab and some previous optimization algorithms for 2 benchmark problems. Some other applications of this tool was introduced in previous works [11-14].
5.1. 10-bar truss problem

Consider a 10 bar plane truss shown in Figure 4 with the following structural characteristics:

- Modulus of Elasticity $E = 10,000$ ksi,
- Material density $\rho = 0.1$ lb/in$^3$,
- Length $L = 360$ in,
- Load $P = 100$ kip.

The structural members are divided into 10 groups. The design variables are the cross section areas of each member group in the interval $[0.1, 35]$ (in$^2$). The constraints are imposed on stresses and displacements. The maximum allowable displacement in the $\pm x$ and $\pm y$ directions for each node is $d_{\text{max}} = 2$ in, while the maximum allowable stress (absolute value) is $\sigma_{\text{allow}} = 25$ ksi in tension or compression and the objective is to minimize the weight of the structure under the specified constraints.

Figure 4. Geometry and applied load for 10-bar truss [11].

| Variables | Optimal cross section area (in$^2$) |
|-----------|-------------------------------------|
| Design name | M. Sonmez [15] | Wu & Tseng [16] | Li et al. [17] | Degertekin & Hayalioglu [18] | Degertekin [19] | Kaveh et al. [20] | This work |
| A1 | 30.548 | 30.378 | 30.704 | 30.429 | 30.208 | 30.5218 |
| A2 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| A3 | 23.18 | 23.468 | 23.167 | 23.244 | 22.698 | 23.1999 |
| A4 | 15.218 | 15.196 | 15.183 | 15.368 | 15.275 | 15.2229 |
| A5 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| A6 | 0.551 | 0.533 | 0.551 | 0.575 | 0.529 | 0.5514 |
| A7 | 7.463 | 7.437 | 7.46 | 7.440 | 7.488 | 7.4572 |
| A8 | 21.058 | 21.084 | 20.978 | 20.967 | 21.189 | 21.559 | 21.0364 |
| A9 | 21.501 | 21.433 | 21.508 | 21.533 | 21.342 | 21.491 | 21.5284 |
| A10 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| Weight (lb) | 5060.88 | 5060.45 | 5060.92 | 5060.96 | 5061.42 | 5062.39 | 5060.9 |
| Number of function evaluations | 500,000 | 32,100 | 125,000 | 16,872 | 7,081 | 9,791 | 347 |

Table 1. Optimization results for 10-bar truss.

Table 1 presents the best optimum results found by the proposed optimization procedure and the corresponding number of function evaluations. The present study is compared with some previous studies found in the literature. It is observed that the optimum weight and design variables obtained from this study are very close to those obtained from previous studies. However, it is clear that the proposed optimization procedure requires much lower structural analyses than other methods to reach the optimum designs.
5.2. 25-bar truss problem

Consider a 25 bar space truss shown in Figure 5 with the following structural characteristics:

- Modulus of Elasticity $E = 10,000$ ksi
- Material density $\rho = 0.1$ lb/in$^3$

The structural members are divided into 8 groups. The design variables are the cross section areas of each member group in the range $[0.01, 5]$ (in$^2$). The constraints are imposed on stresses and displacements. The maximum allowable displacement in the $\pm x$, $\pm y$ and $\pm z$ directions for each node is $d_{\text{max}} = 0.35$ in. Two load cases have been considered. The maximum and minimum allowable stress is shown in the Table 5 of the reference [11]. The objective is to minimize the weight of the structure under the specified constraints for both load cases simultaneously. The members were grouped as follows: (1) element 1; (2) elements 2, 3, 4 and 5; (3) elements 6, 7, 8 and 9; (4) elements 10 and 11; (5) elements 12 and 13; (6) elements 14, 15, 16, 17 and 18; (7) elements 18, 19, 20 and 21; (8) elements 22, 23, 24 and 25.

Table 2 presents the optimum results of the proposed optimization procedure and compares these results with those previously reported in the literature. The difference among all the results is very small, hence all optimization algorithms indicated in Table 2 found almost the same optimum structural weight. It is apparent that the optimization procedure proposed in this work can reach optimum results much faster than the other algorithms.

![Figure 5. Geometry and applied load for 25-bar truss [11].](image)
Table 2. Optimization results for 25-bar truss.

| Variables | Design variables | Optimal cross section area (in²) | M. Sonmez [15] | Li et al. [17] | Degertekin & Hayalioglu [18] | Degertekin [19] | Kaveh et al. [20] | This work |
|-----------|------------------|---------------------------------|----------------|---------------|-----------------------------|----------------|------------------|----------|
| No        |                  |                                 |                |               |                             |                |                  |          |
| 1         | A1               |                                 | 0.011          | 0.01          | 0.01                        | 0.01           | 0.01             | 0.01     |
| 2         | A2-A5            |                                 | 1.979          | 1.970         | 2.071                       | 2.074          | 1.9907           | 1.9856   |
| 3         | A6-A9            |                                 | 3.003          | 3.016         | 2.957                       | 2.961          | 2.9881           | 2.9969   |
| 4         | A10-A11          |                                 | 0.01           | 0.01          | 0.01                        | 0.01           | 0.01             | 0.01     |
| 5         | A12-A13          |                                 | 0.01           | 0.01          | 0.01                        | 0.01           | 0.01             | 0.01     |
| 6         | A14-A17          |                                 | 0.69           | 0.694         | 0.6891                      | 0.691          | 0.6824           | 0.679    |
| 7         | A18-A21          |                                 | 1.679          | 1.681         | 1.6209                      | 1.617          | 1.6764           | 1.6769   |
| 8         | A22-A25          |                                 | 2.652          | 2.643         | 2.6768                      | 2.674          | 2.6656           | 2.6676   |
| Weight (lb) |                 |                                 | 545.193        | 545.19        | 545.09                      | 545.12         | 545.164          | 545.166  |
| Number of function evaluations |                  |                                 | 300,000        | 125,000       | 15,318                      | 9,051          | 13,326           | 851      |

6. Conclusion

This paper demonstrates the efficiency of the Abaqus2Matlab toolbox for structural optimization purposes. Through this paper, a general description of Abaqus2Matlab toolbox and guidelines for performing a general structural optimization procedure using this toolbox are presented. By using this optimization procedure, many structural optimization problems can be solved in a short time with high accuracy. In order to prove the efficiency of this toolbox, two benchmark optimization problems are solved and their results compared to corresponding data published in the literature with a good agreement.

Acknowledgements

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 107.01-2019.322. In addition, this research is supported by Research Fund of Vietnam Maritime University.

References

[1] C.V. Camp, B. J. Bichon, S. P. Stovall. Design of Low-Weight Steel Frames Using Ant Colony Optimization. Journal of Structural Engineering 2005; 131: 369-379.
[2] S. Rajeev, C.S. Krishnamoorthy. Discrete Optimization of Structures Using Genetic Algorithms. Journal of Structural Engineering 1992; 118: 1233-1250.
[3] K.S. Lee and Z.W. Geem. A new structural optimization method based on the harmony search algorithm. Computers & Structures 2004; 82: 781-798.
[4] R.E. Perez and K. Behdinan. Particle swarm approach for structural design optimization. Computers & Structures 2007; 85: 1579-1588.
[5] R. Balling. Optimal steel frame design by simulated annealing. Journal of Structural Engineering 1991; 117: 1780-1795.
[6] V.H. Truong, P.C. Nguyen, S.E. Kim. An efficient method for optimizing space steel frames with semi-rigid joints using practical advanced analysis and the micro-genetic algorithm. Journal of Constructional Steel Research 2017; 128: 416-427.
[7] V.H. Truong and S.E. Kim. An efficient method for reliability-based design optimization of nonlinear inelastic steel space frames. Structural and Multidisciplinary Optimization 2017; 56(2):331-351.
[8] V.H. Truong and S.E. Kim. Reliability-based design optimization of nonlinear inelastic trusses
using improved differential evolution algorithm. *Advances in Engineering Software* 2018; 121:59-74.

[9] S.E. Kim and W.F. Chen. Practical advanced analysis for braced steel frame design. *Journal of Structural Engineering* 1996; 122: 1266-1274.

[10] G. Papazafeiropoulos, M. Muñiz-Calvente, E. Martínez-Pañeda. Abaqus2Matlab: a suitable tool for finite element post-processing. *Advances in Engineering Software* 2017;105:9–16.

[11] Q.V. Vu, G. Papazafeiropoulos, C. Graciano, S. Kim, Optimum linear buckling analysis of longitudinally multi-stiffened steel plates subjected to combined bending and shear, *Thin Walled Struct.* 136 (2019) 235–245.

[12] Q.V. Vu, V. Truong, G. Papazafeiropoulos, C. Graciano, S. Kim, Bending-buckling strength of steel plates with multiple longitudinal stiffeners, *J. Constr. Steel Res.* 158 (2019) 41–52.

[13] V.H. Truong, G. Papazafeiropoulos, V.T Pham, Q.V. Vu. Effect of multiple longitudinal stiffeners on ultimate strength of steel plate girders. *Structures* 22 (2019) 366–382.

[14] G. Papazafeiropoulos, Q.V. Vu, V.H. Truong, M.C. Luong, V.T. Pham. Prediction of buckling coefficient of stiffened plate girders using deep learning algorithm, *CIGOS 2019, Innovation for Sustainable Infrastructure, Lecture Notes in Civil Engineering* 54,1143-1148.

[15] Mustafa Sonmez. Artificial Bee Colony algorithm for optimization of truss structures. *Applied Soft Computing* 2011; 11(2): 2406-2418.

[16] C.Y. Wu and K.Y. Tseng. Truss structure optimization using adaptive multi-population differential evolution. *Structural and Multidisciplinary Optimization* 2010; 42: 575-590.

[17] L.J. Li, Z.B. Huang, F. Liu, Q.H. Wu. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers & Structures* 2007; 85: 340-349.

[18] S.O. Degertekin and M.S. Hayalioglu. Sizing truss structures using teaching-learning-based optimization. *Computers & Structures* 2013; 119: 177-188.

[19] S.O. Degertekin. Improved harmony search algorithms for sizing optimization of truss structures. *Computers & Structures* 2012; 92-93: 229-241.

[20] A. Kaveh, T. Bakhshpoori, E. Afshari. An efficient hybrid Particle Swarm and Swallow Swarm Optimization algorithm. *Computers & Structures* 2014; 143: 40-59.