RESEARCH REPORT

VTT-R-00998-14

SIMPRO

Parameter Optimisation Using Heeds MDO and DAKOTA

Authors: Pekka Rahkola

Confidentiality: Project confidential (will be public after the project)

Report's title

Parameter Optimisation Using Heeds MDO and DAKOTA

Customer, contact person, address	**Order reference**
Tekes Matti Säynätjoki Kyllikinportti 2, P.O. Box 69, FI-00101 Helsinki, Finland	
Project name	**Project number/Short name**
Computational methods in mechanical engineering product development	78634/SIMPRO
Author(s)	**Pages**
Pekka Rahkola	29/-
Keywords	**Report identification code**
Elevator, modelling, multibody, optimisation, simulation	VTT-R-00998-14

Summary

The objective of the work was to study the process of the parameter optimisation applied to a multibody simulation model and to run two optimisation cases using two optimisation software applications: HEEDS MDO 7.1 and DAKOTA version 5.3.1.

The industrial case studies were performed using a simulation model of an elevator. The selected vibration feature was optimised using different design variables of the system. The feature was evaluated using a scalar valued objective function and also inequality constraints for the optimisation were defined.

Both optimisation case studies were successful. Both case studies were run with the default parameter values of the optimisation methods. Tools for the parameter optimisation and especially the mathematical algorithms inside them were found capable and robust for parameter optimisation of multibody models.

Confidentiality	Project confidential (will be public after the project)

Espoo 27.2.2015

Written by	**Reviewed by**	**Accepted by**
Pekka Rahkola, Senior Scientist	Kai Katajamäki, Principal Scientist	Johannes Hyrynen, Head of Research Area

VTT's contact address

VTT Technical Research Centre of Finland, P.O. Box 1000, FI-02044 VTT, Finland

Distribution (customer and VTT)

Tekes, 1 copy
VTT, 1 copy

Contents

1. Introduction

The use of modelling and simulation, and more widely the use of computational models, is common within engineering work. Typical applications of modelling and simulation are related to physical systems, for example structural dynamics and vibration of a mechanical system or electromagnetics of electrical systems. The main benefits of the modelling and simulation are the cost efficiency, when the need to build physical prototypes is smaller, shorter time to market, when more information of the product can be gathered faster and in the earlier phase of the design process, and even safety, when testing can be done virtually. The use of computational models gives valuable information of the system without the need to build and test the system in real life.

In engineering work the making of design choices often lead to the making of compromises, when design aspects are competing against each other. For example, the performance of the system should be maximised while the mass of it should be minimised. This sets the requirement to find optimal designs in a systematic way. The increase of computational power has made it possible to simulate extensive analysis rather than only limited design cases [1]. The use of optimisation needs the coupling between the simulation software and the optimisation process, the handling of the iterative optimisation process, and management of the models and results. Usually the sensitivity of the system between the optimised design variables system behaviour and response is also studied. This can be done using design of experiment (DOE) studies.

In mechanical engineering the main subfields of optimisation are the parameter optimisation, and the topology optimisation. In parameter optimisation the performance of the system is optimised by varying design variables of the system inside a defined range. In topology optimisation the layout of the material in a pre-defined space is optimised against a given set of loads, boundary conditions, and other design constraints. This work focuses on parameter optimisation applied to a mechanical system.

2. Objective

The objective of the work was to study the process of the parameter optimisation and to run two optimisation cases using two optimisation software applications. In this report the optimisation process is first described in general by defining the main components and functions of the optimisation problem and formulating it in a mathematical form. After that two optimisation software case studies using HEEDS MDO 7.1 and DAKOTA version 5.3.1 are introduced. The codes are tested with an optimisation run applied to a multibody simulation model of an elevator. In the conclusion section the main steps of using parameter optimisation in engineering work are discussed.

3. Optimisation Process in General

The main components of an optimisation process are the target system, design variables with their limits, objective functions and constrain functions, and optimisation algorithm. In a typical case the target system is a computational model of a system to present its behaviour. An example is a multibody model for a dynamic analysis of a mechanical system. Design parameters are parameters of the target system; for example mass of a body, distance between connection points, or the stiffness of a spring component. Design variables together with their limits define the design space for the optimisation process. Design variables have an effect on the target system's behaviour, which is measured using the objective functions. In a case of a multibody simulation of a mechanical system, the objective function is usually displacement, velocity, or acceleration response of the system. Constraint functions are also measures of the system, which judges the feasibility of the design inside the design space.

An example of a constraint function could be the mass of a part, which has the maximum value defined, or the displacement response of a part, which has also the maximum value defined. The optimisation algorithm is the logic which controls the iterative optimisation process. It defines the values for the design variables during the process based on simulation process evaluations.

The purpose of the optimisation process is to minimise or maximise one or several objective functions by changing a set of design parameters inside a design space and without violating design constraints. In a mathematical form the optimisation problem can be formulated as follows: minimise (or maximise) the objective function

$$F(\boldsymbol{b}) = F(\boldsymbol{b}, t), \qquad t \in [t_0, t_1] \tag{1}$$

using a set of design variables $\boldsymbol{b} \in R^n$ inside the design space

$$b_{i,min} < b_i < b_{i,max}, \qquad i = 1, \dots, n. \tag{2}$$

At the same time inequality constraints (also equality constrains may be used) needs to be fulfilled,

$$g_j(\boldsymbol{b}, t) < c_j, \qquad \forall t \in [t_o, t_1], j = 1, \dots, m. \text{ [2]} \tag{3}$$

The objective function $F(\boldsymbol{b})$ can be a scalar-valued function or a vector-valued function. There are three basic types for objective functions [2]:

- Response independent; the value for the objective function is defined for example based on geometry or other design variables.

- Time point; the value for the objective function is defined at specific time point from the time series, for example a peak-to-peak value on acceleration time series. This is the typical case for a simulation model optimisation, because the responses of the simulation model are functions of time.

- Integral type; the value for the objective function is defined using all time points from the time series, for example an RMS value calculated on acceleration time series.

Usually the shape of the object function is not known unless the target system is for example an analytical function. Depending on the shape of the objective function, it may have zero, one or several minimums or maximums. In a case of multi-objective optimisation there is no unambiguous solution, but the result can be shown as Pareto curves. It means that from the evaluation of the object function point of view, all designs lying on the Pareto curves are as good as others.

Designs inside the design space can be categorised into feasible and infeasible designs based on the constraints $g(\boldsymbol{b})$. An example of a constrain function could be a displacement of a body during the simulation time span $t \in [t_0, t_1]$. If an inequality constraint is defined, a feasible design is met, when the displacement is below the constraint limits. Constraints are also defined using the result time series and they can be scalar or vector functions. The same basic types introduced for object functions are valid for constraints as well [2].

The optimisation process is controlled by the optimisation method. The behaviour of the system defines the requirements for the used method. There are several different approaches, which are selected based on the behaviour of the simulation process and the type of the optimisation. This section will give a short description of the used optimisation methods for continuous design spaces. In gradient based methods, which are known also as deterministic optimisation methods, the next evaluated design during the iterative process is defined using the previous design and a step in a certain direction. This procedure can be defined as an iterative formula

$$b^q = b^{q-1} + \alpha S^q, \quad (4)$$

where the design b^q is defined using the former design b^{q-1} and a step α in a direction of S^q [2]. The search direction is usually based on the gradient information and the solution navigates and converges into the direction of the smallest gradient. There are different ways to estimate the gradient. Well-known examples are for instance finite differences, likelihood ratios, and perturbation analysis [3]. Gradient based methods are suitable and efficient for the local optimisation, but they cannot be used in the global optimisation problems, when the response surface is non-convex. Due to the deterministic nature the parallelization of gradient based methods is difficult.

Global search method is required, when the response surface is non-convex, gradient information is not available or it cannot be estimated reliably. When using deterministic methods in a case of nonlinear system responses, the optimisation may end up in a local minimum and there is no way to guarantee that the optimum is more than local. The basic function with the global methods is to overcome the trap between the local and the global optimum. Stochastic methods, which are based on using random variables, are capable on finding global optimums in large design spaces. Stochastic methods go through the design space coarsely and focus the study on the promising area of the design space. Other suitable approaches for the global optimisation procedures are also methods based on heuristics. Examples of such methods are for instance simulated annealing and evolutionary algorithms [3]. The simulated annealing has some analogy to the annealing process in metallurgy. The initial design is chosen randomly. The next design to be evaluated is chosen from the neighbour. If the objective functions get better, that design will be the current solution design. If the neighbour design is not better, the method will keep that solution with a probability which increases during the iteration. Evolutionary algorithms use mechanisms, which are taken from the biological evolution [4]. These algorithms operate on a population of solution rather than a single solution. Genetic operators are applied to the population to define the new offspring. During the evolution poor designs will extinct and better designs reach towards the optimum.

In general global optimisation methods are computationally more expensive than efficient local methods because they need usually more iterations. On the other hand the capability for the parallelization of optimisation process is better. If the number of iterations is high and at the same time simulation process is computationally expensive, a suitable approach can be Response Surface Methodology. The method is based on a metamodel, which is fast to calculate. The metamodel, which is a regression surface model, is defined using surface fits into several response points defined using the original model. This metamodel can be used to defined the model output with a certain accuracy in a few seconds. The optimisation is done using the regression function and the approach is efficient [4].

The schematic presentation of an optimisation process for a simulation process as the target system is shown in Figure 1. The simulation model has its input files for the execution and baseline values for the design variables. The optimisation process is solved iteratively. When the simulation process is finished, object and constrain functions are evaluated, and the design feasibility is defined. After that the optimisation method defines new values for the design parameters based on the implemented algorithm and runs the simulation process again, if the optimum solution is not yet reached. Based on the method implementation and logic behind the iterated values of the design parameters and the progress of the solution convergence are different.

The success and efficiency of the optimisation can be increased if the behaviour of the optimised system is known beforehand. Typical analyses, which give information about the dynamics of the studied system, are:

- Parameter study to define the effect of one or several design parameters on the system response. This can be used for example to define the sensitivity of the system

relative to the studied design parameters within the design space or around a certain design point.

- Design of Experiments (DOE) to study the effect of a design space on the system response. DOE studies can be used for example to perform a sensitivity analysis, which identifies those design variables that have the most influence on the response [5]. This information is valuable before the optimisation run to define the essential parameters for the study. Another use is to define the system response for a set of design parameters to be used with the response surface methodology (see optimisation method description above).

- Uncertainty quantification analysis to study the robustness and reliability of the system response. In these studies stochastic distributions are applied to design parameters to define the effect of system behaviour against changes in environment or tolerances. As an example the effect of input uncertainty on the response uncertainty can be studied.

These analyses give information about the dynamics of the studied system, which may help to find the optimum design and also gives information of the system behaviour around the optimum solution. These analyses are usually included in optimisation software packages.

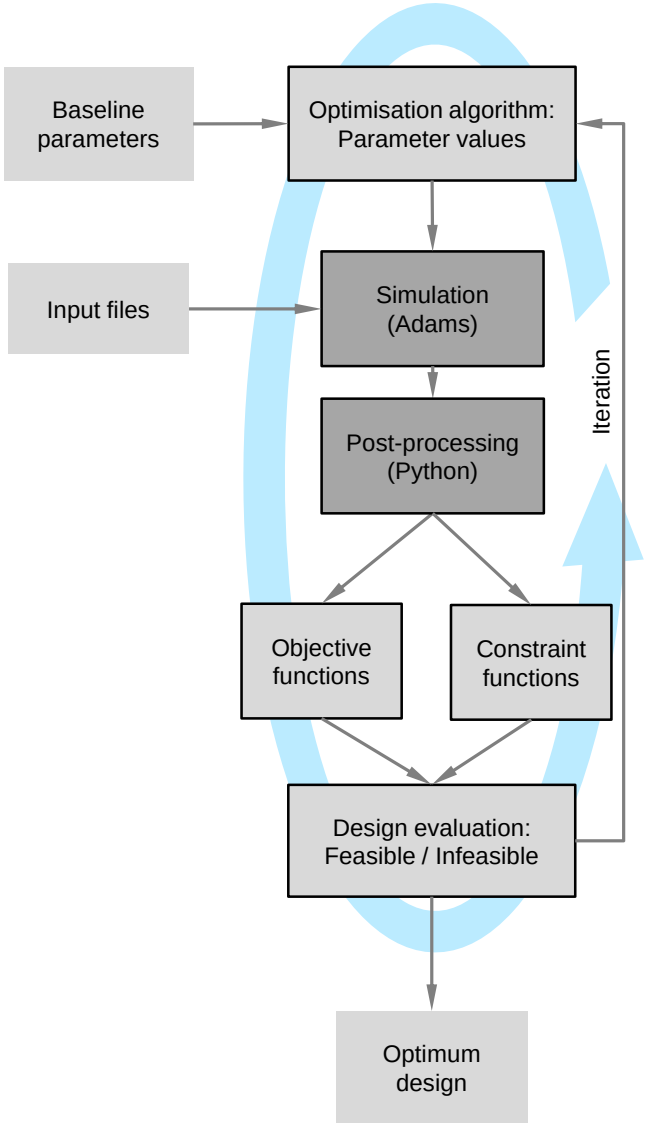

Figure 1. Schematics of an optimisation process for a multibody system simulation model.

4. Simulation Case

In this work the parameter optimisation is applied to a multibody system simulation model. The multibody system approach is suitable for modelling of mechanical systems with large motions and rotations involved. Multibody models consist from bodies, which are connected together using idealised joints or force elements like springs and dampers. The properties of the bodies are defined by the geometry (mass and inertia), the joints by the location, orientation and degree of freedom information and force elements by the spring stiffness and damping values etc. The bodies may be considered as rigid or deformable if it is has effect on the dynamics of the system. [1]

In this work the object for the optimisation study is a multibody system simulation model of an elevator. The elevator model of the example case is shown in Figure 2. The simulation model, which is described in [5], is developed using MSC.Adams 2013.1 software. Main assemblies of the elevator model are the frame structure, the platform, and wheels with supporting structures. All bodies are modelled as rigid. The model has total 36 degrees of freedom. The duration of the simulation case is about 4 minutes.

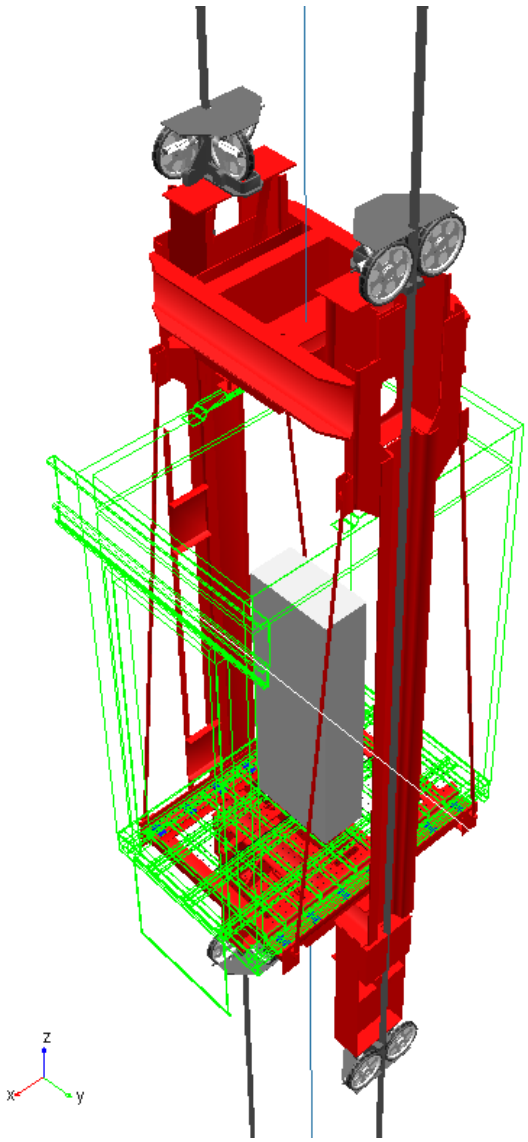

Figure 2. Multibody simulation model of the studied elevator.

There are two different test cases for optimising the selected vibration feature of the elevator described in this report. The needed variables of the simulation model are parametrised using design variables. The responses of the model for the elevator behaviour evaluation are displacements and accelerations. The objective function for the studied time dependent optimisation process consists of integral and point type of components. In the studied cases a scalar optimisation is used: the sum of objective function components is minimised. All objective values are non-negative and independent. Thus minimising the sum of them is reasonable and there is no need for the use of multi-objective optimisation. Both optimisation cases include also nonlinear constraint functions for certain responses, which needs to be under certain boundaries.

5. Case Study Using HEEDS MDO

Heeds MDO is design optimisation software from Red Cedar Technology[1]. Software is suitable for scalar and multi-objective design parameter optimisation with multiple constraints, design of experiments (DOE) studies, and robustness and reliability studies, which cover sensitivity analysis using stochastic variables and constraint violation studies [7]. In a graphical user interface HEEDS MDO has tools to create and manage the iterative calculation process and to view the results. Important product features of HEEDS MDO are the capability to parallel computation, and the use of portals, which are ready defined interfaces to establish HEEDS MDO processes for commonly used modelling and simulation tools. Portals are currently available for example to Adams, Abaqus, Ansys, Nastran, Excel, Matlab, and Python.

In HEEDS MDO a new study, which can be an optimisation, DOE or robustness and reliability study, is called a project. The project contains one or several processes, which can be for example simulation using FE code or a post-processing using Python script. A process is defined using input and output files and the execution command with its command options. Processes have design variables as inputs and responses as outputs. Design variables of the project are connected to the input files of processes and responses are connected to output files of processes. Processes can be connected together and outputs of one process can be used as inputs of another process. The study in the project is defined using the design variables and responses. For example in an optimisation study, design variables of the project are connected to parameters of the optimised model and objectives and constrains are defined using model's responses.

5.1 Preparing the Simulation Model for the Parameter Optimisation

The easiest way to define a process in HEEDS MDO is to use a portal. There is a portal for Adams available in HEEDS MDO and it will be used in this study. Adams portal allows the user to create design variables and responses for the HEEDS MDO process directly from the design variables and objective components defined in the Adams model. As an input file for the Adams portal only the binary formatted Adams database file is defined. Before the definition of HEEDS MDO process using Adams portal can be started, there are few steps, which have to be done to the Adams model. The steps needed for the Adams model to prepare it for the optimisation run are described in the following.

First the simulation model in Adams must have design variables and design objectives to be used as design variables and responses in the HEEDS MDO process. Measures of the Adams model to be used as responses in HEEDS MDO process are defined using *Design Objectives*. An example of a design objective creation is shown in Figure 3. The design objective is defined using a request component and it stores the displacement measurement during the simulation. After the creation of design objectives the model needs a simulation script to define the simulation case. In this case study the simulation script contains a static equilib-

[1] Red Cedar Technology: http://www.redcedartech.com/

rium analysis in the beginning and then dynamic analysis (see Figure 4). As a last step for the use of Adams database file with HEEDS MDO, the full name option in Adams/View has to be used. It can be set from *Settings → Names → Full names*.

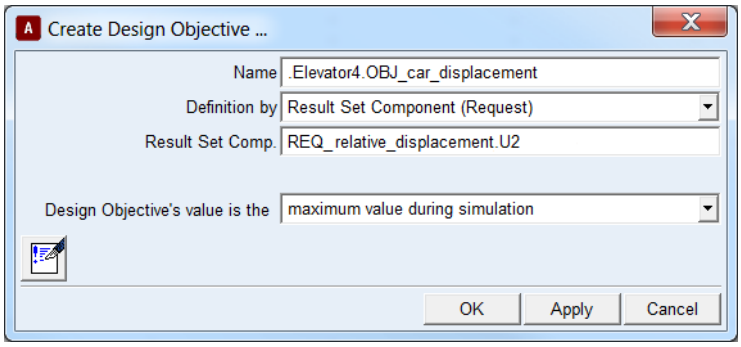

Figure 3. The creation of a design objective in Adams/View.

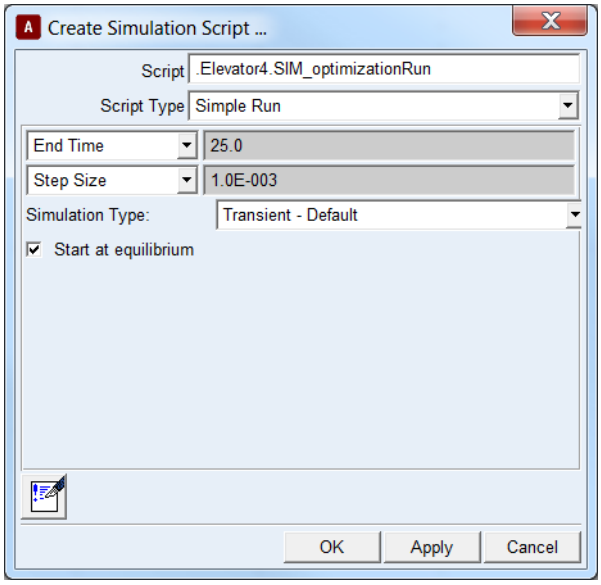

Figure 4. The creation of a simulation script in Adams/View.

In the studied case the simulation model requires input files for the tire model. The location for the input files in the model can be given as a path relative to the model directory or as an absolute path. In this case paths to these input files are given as absolute paths in the Adams model. The benefit of this approach is that the model can be simulated in any folder and the input files are always accessible, without the need to copy the input files with the model file. This has more importance in an iterative optimisation process with hundreds or thousands of evaluated designs. If the place for the input files would be given as a path relative to the model file, all input files needs to be copied to the directory where the simulation is run. An alternative way to do this in HEEDS MDO would be to use an option in the preferences to run the simulations a fixed model directory. Then there would be no need to copy neither the input files nor the model file, only case results.

When using the Adams portal in HEEDS MDO, the simulation is run in Adams/View batch mode. To pass information of the simulation progress or error messages in a case of simulation failures to HEEDS MDO, the log file written by the Adams/View can be used. The option for the solver to write the message file is found from *Settings → Solver settings → Display → Show Messages → Yes* (see Figure 5). By default only warnings and errors are written to the log file. To get information of the progress of the solving process, the severity level of message settings should be changed to the information level (see Figure 6) from *Message window → Settings → Message settings → Severity level → Information*.

For the use of Adams portal in HEEDS MDO, two additional files beside the Adams database file are needed. The first one is an xml file containing a description of model design variables and design objectives. The xml file is created by the HEEDS plugin. The second is a text file with a2h extension, which contains the name for Adams database file and execution command for the Adams run. All these files are exported from Adams using the plugin manager's HEEDS Export button. The export dialog asks the model name, the name for the exported files and the simulation script to be used (see Figure 7). After these steps the simulation model defined in Adams/View is ready to be used in HEEDS MDO via Adams portal. To make the HEEDS Export plugin work three files, *HEEDS_plugin.bin*, *HEEDS_plugin_plg.xml*, and *MDOIcon-48x48-32.xpm*, coming with the HEEDS MDO installation was copied into given place of the Adams installation:

```
C:\MSC.Software\Adams_x64\2013_1\Win64
```

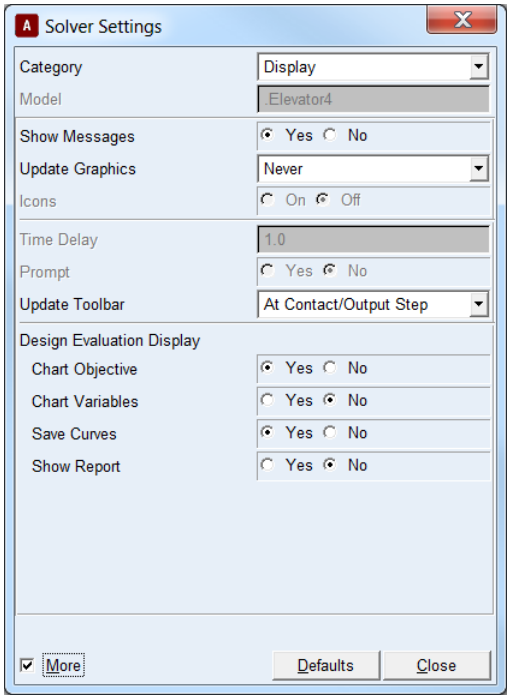

Figure 5. The modification of solver settings to show simulation messages.

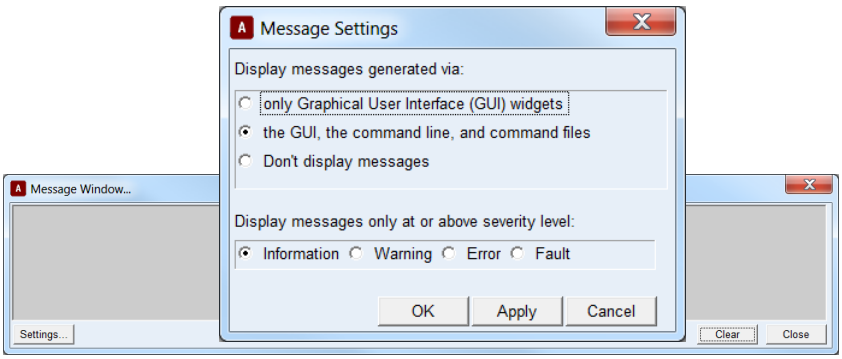

Figure 6. The modification of the message setting's severity level.

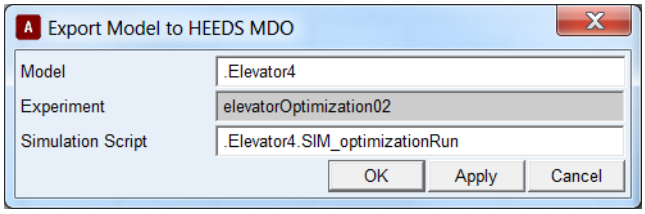

Figure 7. The HEEDS export dialog in Adams/View.

5.2 Defining the Project

In this case study the optimisation project in HEEDS MDO contains two processes: simulation using Adams/View, called *AdamsRun*, and post-processing using a Python script, called *PostProcess* (see Figure 8). There is a connection from the *AdamsRun* to the *PostProcess*, which defines the execution order of the processes in the iteration loop. In this case *AdamsRun* is executed first and after it is finished *PostProcess* is started.

AdamsRun process is set up using the Adams portal. After selecting the Adams portal during the project definition, the execution command and command options for Adams becomes visible (see Figure 8). The execution command for Adams is *mdi.bat*, which is found from the Adams installation directory. In this case study the execution command in Windows environment is

```
C:\MSC.Software\Adams_x64\2013_1\common\mdi.bat
```

Default command options of the Adams portal were ready to use. Default command options are done using HEEDS MDO variables *%APPNAME%* and *%ANALYSIS%*, which are replaced with strings during the design evaluation (see Figure 8). The execution command for Adams after string replacement is

```
aview ru-st b HEEDS_updata_AdamsRun.cmd exit
```

which tells that Adams/View (*aview*) is run using as a standard version (*ru-st*) in batch mode (*b*) and the command file (*HEEDS_update_AdamsRun.cmd*) is loaded [8]. The command file that is read is created automatically by HEEDS MDO. It contains Adams/View commands to read the binary formatted Adams database file, modify the design parameters of the model, run the simulation script, evaluate the design objectives, and write them into a file.

When using the Adams portal the input and output file needed for the process is the binary formatted Adams database. In addition to the database file also files with extensions .a2h and .xml, which are exported from Adams/View using HEEDS plugin (see Chapter 5.1), has to be in the same location as the database file.

To make sure that the simulation run using Adams/View has finished successfully during the optimisation process, a success check needs to be defined. There are several options for the success check definition available in HEEDS MDO. In this case the check is done by searching a string "*finished*" from the log file written by Adams/View (see Figure 9). If the simulation using Adams/View has stopped due to an error, then the "*finished*" string is not found. In this case HEEDS MDO will mark that design as an error design and results of it are not taken into account in the project. The use of success check is important, because it is the only way for HEEDS MDO to know that the simulation or any other process has ended successfully. Without any success check a failed simulation with some results files existing can lead the optimisation process in a wrong direction. At the moment only one success check can be defined for one process. Sometimes there would be also a need for several success checks, for example separate checks for a static equilibrium analysis and a dynamic analysis in a

case of an Adams/View run. If the static analysis fails, the dynamic analysis may still be successful, but the results may not be sufficient.

The second analysis in the process is called *PostProcess*, which executes a Python script. There are three input files and one output file needed for the *PostProcess* (see Figure 10):

- *adamsRun.res* Input Result file saved from the Adams run
- *pyPostProcess.py* Input Python script for the post-processing
- *runPostProcess.bat* Input Batch file for running the Python script
- *postProcess.dat* Output Result file of the post-processing

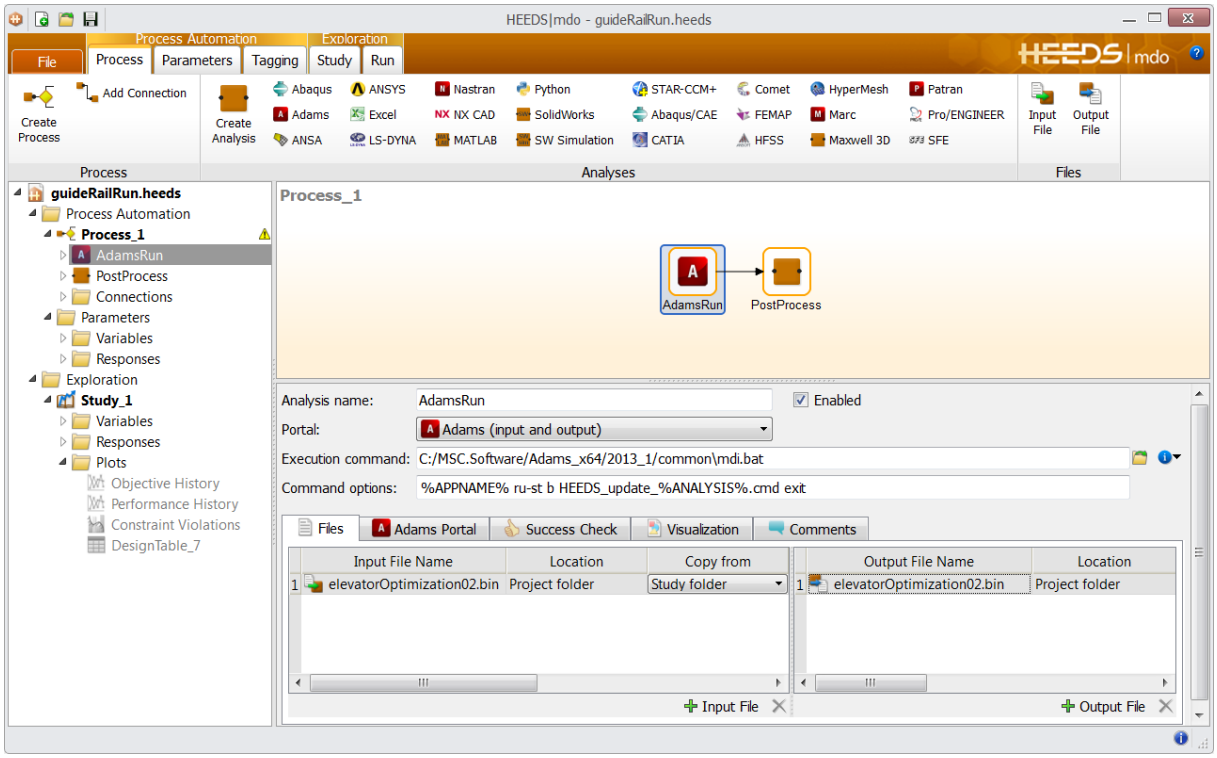

Figure 8. The usage of the Adams portal in HEEDS MDO.

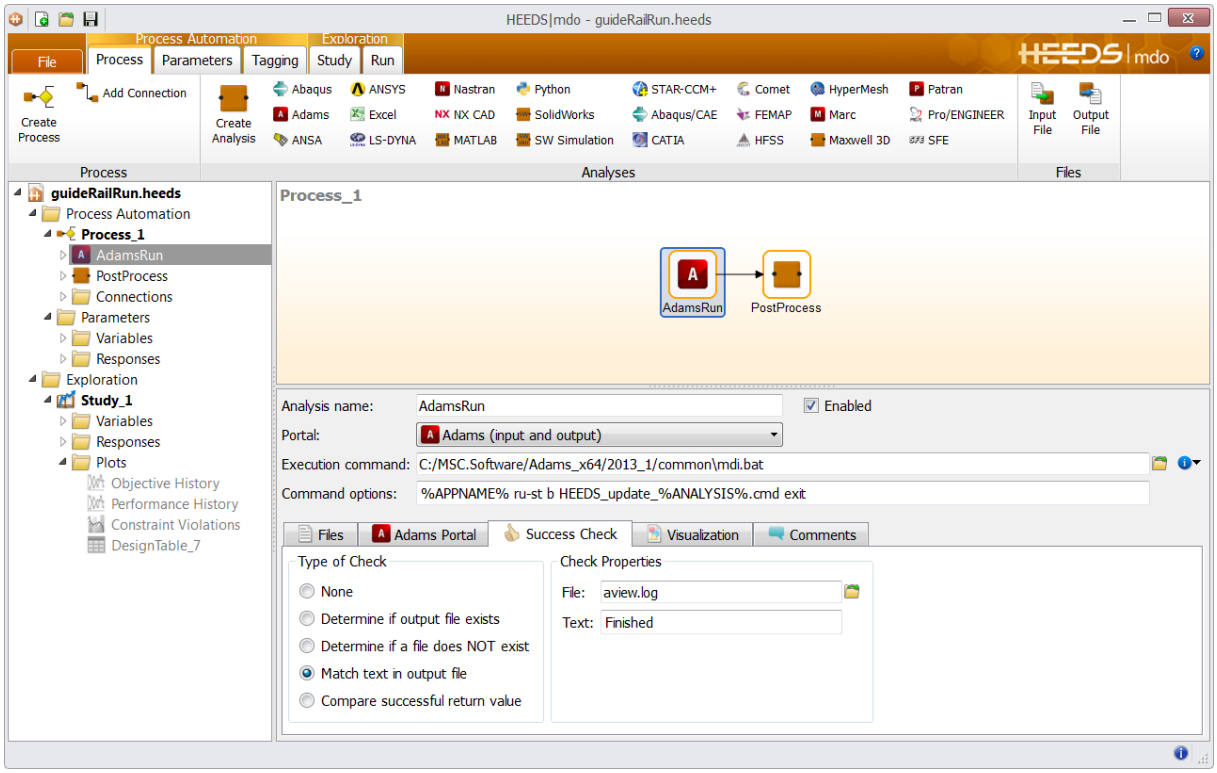

Figure 9. The definition of the success check for the Adams run in HEEDS MDO.

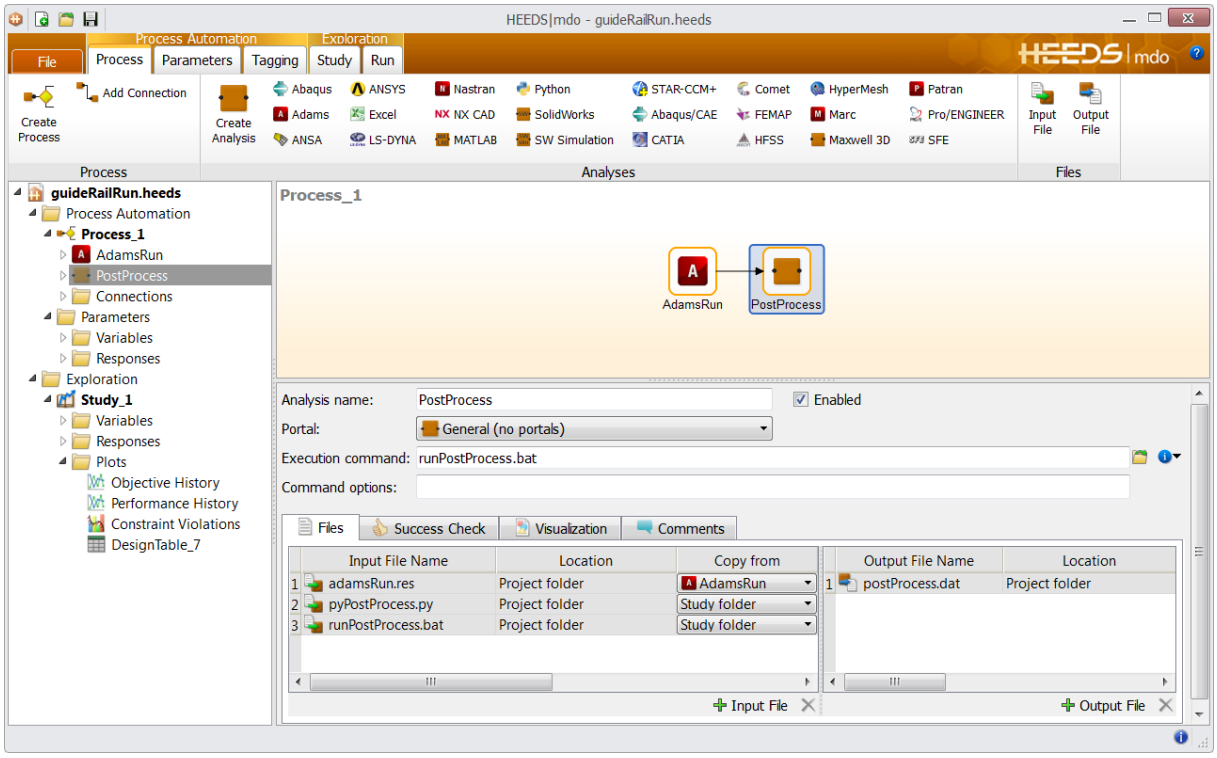

Figure 10. Definition of the process called PostProcess in HEEDS MDO.

The execution command for the *PostProcess* analysis is the batch file *runPostProcess.bat*, which calls Python and runs the script given as an input file. The script reads the result file of the Adams run, calculates the objective function values of the result time series, and saves them into the output file *postProcess.dat*.

PostProcess analysis is built with the general portal even there would have been a portal and an installation for Python available in the HEEDS MDO. The reason for not to use Python portal was that the *PostProcess* script needs some additional libraries, for example *NumPy*, which aren't included in the default HEEDS MDO installation. To be able to run another Python installation instead of Heeds' one, environment variables pointing to HEEDS MDO's Python has to be unset. This is done in *runPostProcess.bat* file.

After importing input and output files for the analyses, project parameters, variables and responses, has to be defined and connected to these files. The type of variables can be continuous, discrete, stochastic, dependent, or constant. The phase of connecting design variables and responses to the input and output files is called tagging. When using the Adams portal, the tagging can be done using automatic tagging wizard, which is started right after importing the Adams input file. The wizard gives a list of all design variables of the Adams model also from the sub model levels. The user selects the needed design variables and HEEDS MDO creates automatically variables of those. The wizard is suitable for one dimensional design variables, but the disadvantage is that it cannot handle vector variables. Design objectives defined in the Adams model are also found by HEEDS MDO and these can be tagged as responses. After that responses are defined automatically.

In a case of using the general portal for process definition, which is valid for *PostProcess* analysis, variables and responses have to be defined and tagged manually. In this case study there are four response values connected to *PostProcess* analysis, which are found from the output file *postProcess.dat*. The output file is a delimited text file and using the option *Delimited* and correct delimiters on the Tagging tab the output file is opened and parsed correctly and the appropriate cells can be selected. After the tagging the response names are filled to the cells.

5.3 Defining the Study

After the needed analyses are defined and the parameters and responses are created and tagged for the project, the study is ready to be defined. The study can be an optimisation study, a DOE study, a robustness and reliability study, or a direct evaluation. Each study type has its own individual definitions. In this case study a scalar optimisation is used. The definition of this study contains variables, responses, and the solving method. Variables and responses of the study are selected from the project parameters. For the variables proper limits according to the optimisation case needs to be defined. Objectives and constraints of the optimisation are selected from the responses of the project. In this case study a scalar optimisation with four objectives and eight constraints is used. For a scalar optimisation case the weighted sum of all objectives is used.

The method searches iteratively the optimum point by changing the tagged variables and calculating the response values and checking the constraint violations. There is a large variety of methods available for scalar optimisation in HEEDS MDO. The selected method for the studied case is *SHERPA*, which is also the default search algorithm in HEEDS MDO. *SHERPA* method is a hybrid method which combines both global and local search methods [3]. In HEEDS MDO the designs are ranked during the search using a performance value which takes into account the objective values and constraint violations. For an infeasible design which doesn't satisfy all constraints, constraint violations give penalty and degrade the performance. The performance rating of a design is defined using values of objectives O_i, constraints C_i, corresponding normalization factors $O_{norm,i}$ and $C_{norm,j}$, weighting factors $O_{wght,i}$ and $C_{wght,j}$, and S_i, which defines the sign for the objectives; for objectives to be minimised it is -1 and for objectives to be maximised it is 1. In a case of n objectives and m constraints the performance value of the design is given by equation

$$Performance = \sum_{i=1}^{n} \frac{O_{wght,i} S_i O_i}{O_{norm,i}} - \sum_{j=1}^{m} \frac{C_{wght,j} g_j^2}{C_{norm,j}^2}, \quad (5)$$

where a constraint violation g_j is defined by

$$g_j = \begin{cases} C_{norm,j} - C_j & , C_j > C_{norm,j} \\ 0 & , C_j \leq C_{norm,j} \end{cases}. \text{[3]} \quad (6)$$

The user can set the values for weighting factors, and normalization factors. In this study default values for the weighting factors were used: $w_{O,i}$ for all objectives are unity and $w_{C,j}$ for all constraints are 10000. As normalization factors $O_{norm,i}$ objective values from the baseline design were used, and as normalization factors $C_{norm,j}$ constraint limits were used. For the studied case the use of the objective normalization sets the importance level of all objectives during the optimisation the same.

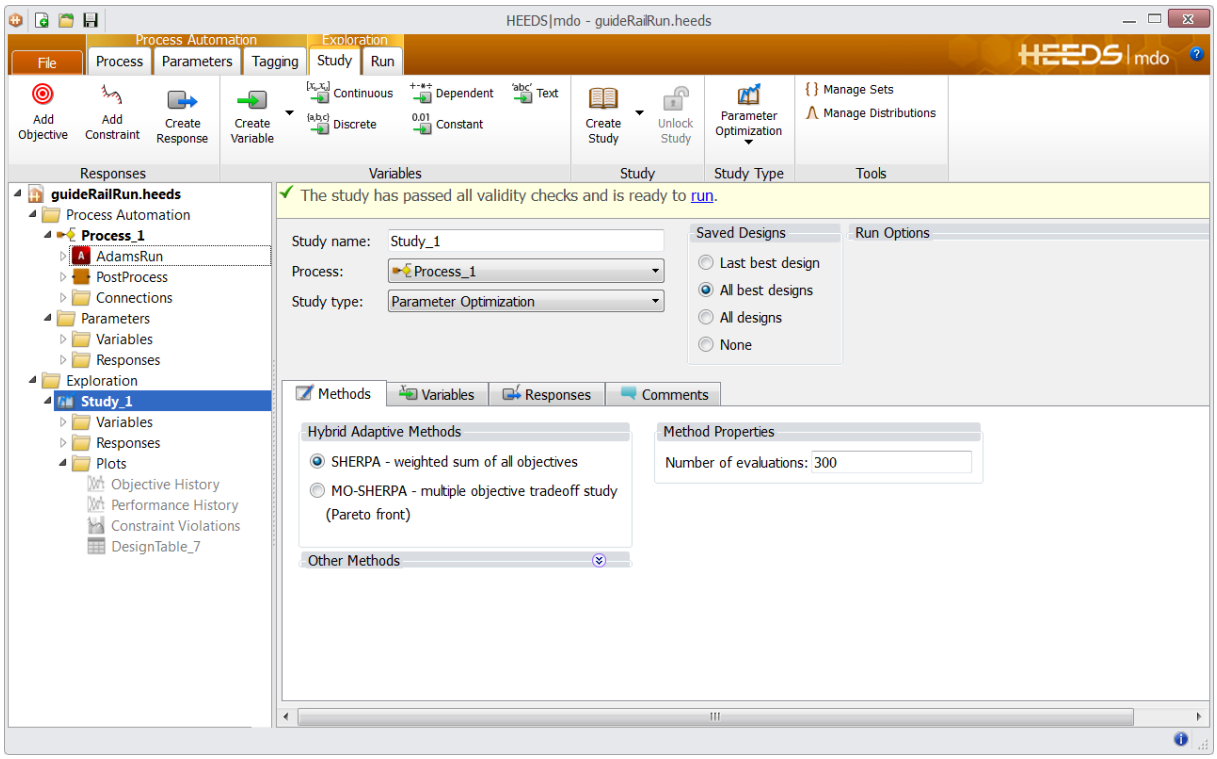

Figure 11. The definition of an optimisation study.

5.4 Viewing the Results

In the studied case the default optimisation method called SHERPA was used. The maximum number of evaluations was set to 400. The method was found robust and capable of converging to a global optimum for the tested case. The history plot for the design performance of the optimisation run is shown in Figure 12. The baseline design is shown as a grey box in the lower left corner, and the best design as a yellow circle. The rate of the change for the performance value is high for the first 150 design and it starts to slow down after that. The derivative of the performance value reaches towards zero when design id reaches 400. This means that the potential improvement on the performance is achieved and the number of design evaluations is high enough.

The second plot in Figure 13 shows the objective values as a function of performance value for feasible designs. The plot shows the change of objective values, when the designs get better. In this case both objectives have got better relative to the baseline value. It can be

seen that there are individual objective values which are lower than the values of the optimum point. However the sum of all objectives defines the performance of the design.

Figure 14 shows an example of a parallel plot which contains the view for relationship of variables and responses of all designs at the same time. Parallel plots can be used to give an overview of the case behaviour. The parallel plot highlights the baseline design and the best design and shows the limits used for constraints.

Figure 15 shows an example of a constraint violation plot for the study. It shows the number of individual constrain violations during the study. It can be used to define what violations are the most critical and hardest to fulfil.

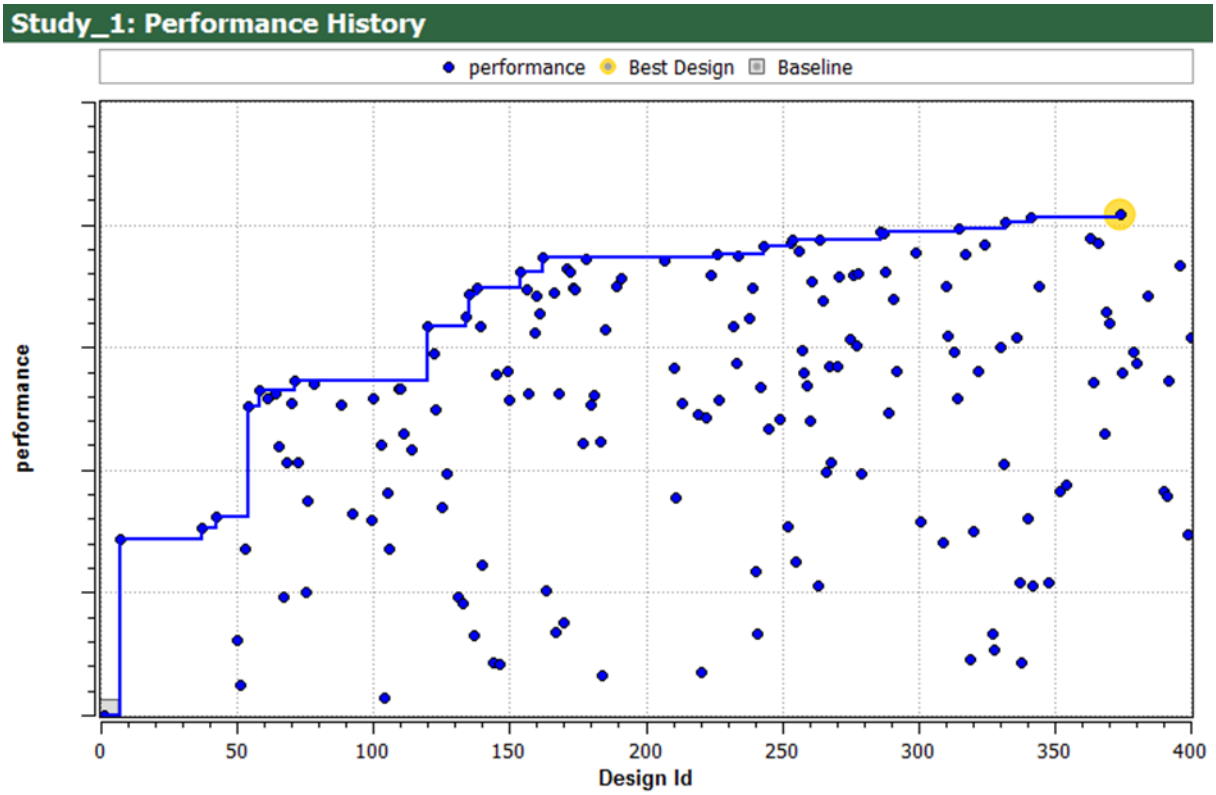

Figure 12. The performance value as a function of design id.

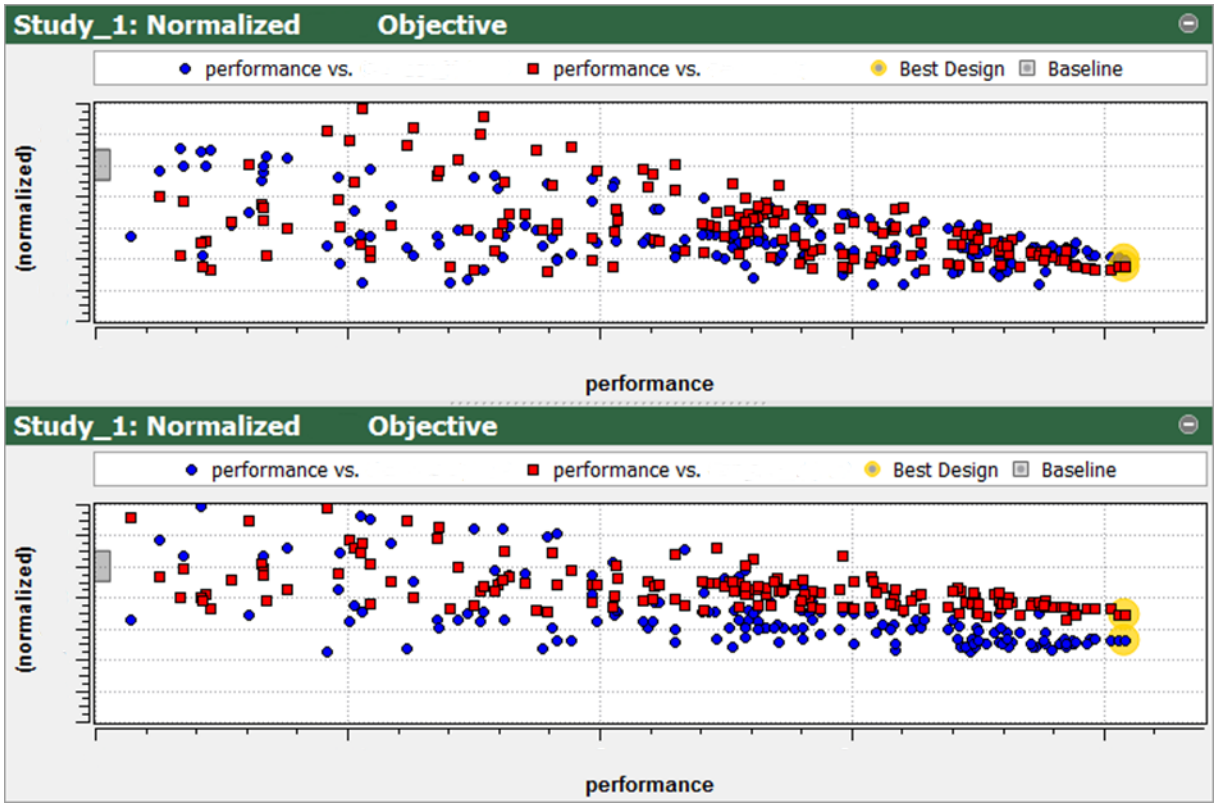

Figure 13. The objective values as a function of performance value for feasible designs. The baseline design is visualised as the grey box in the left side and the best designs are visualised with the yellow circles.

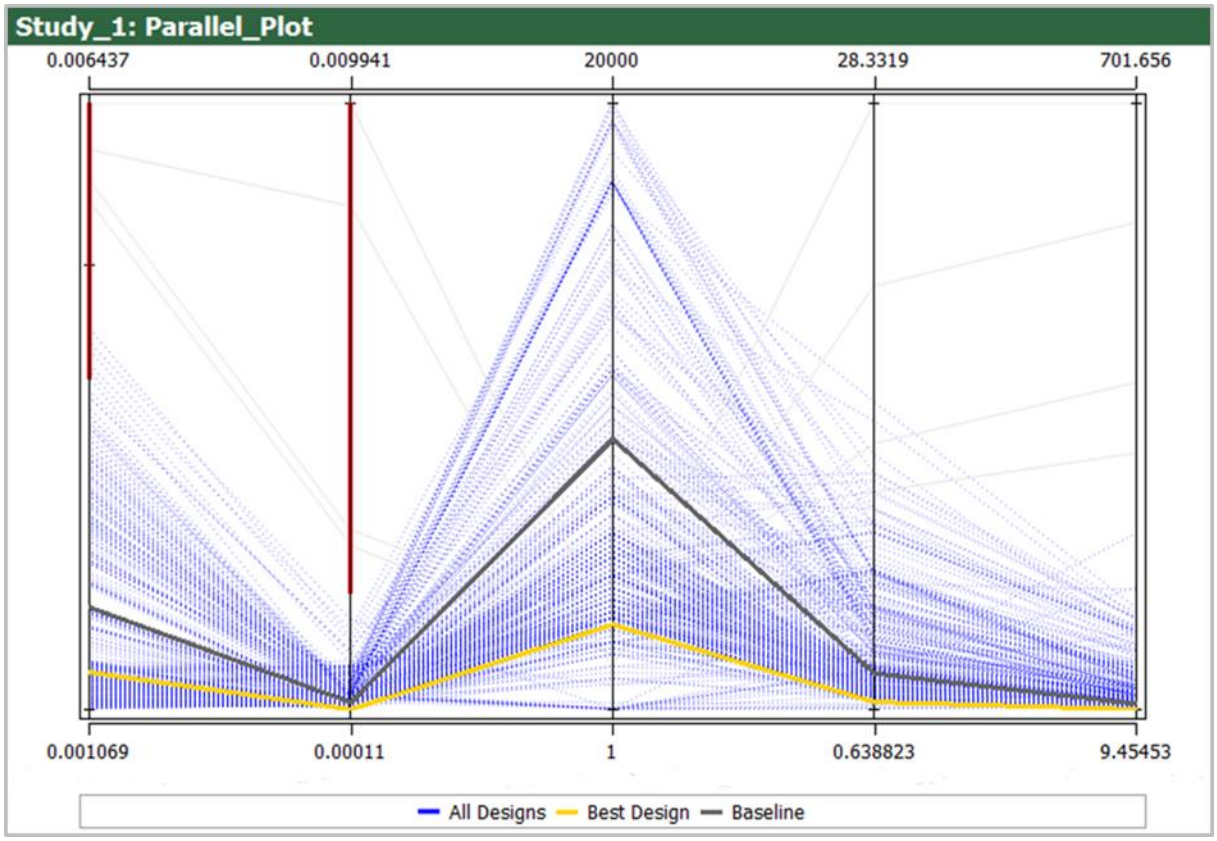

Figure 14. An example of a parallel plot for five design parameters and their constraints.

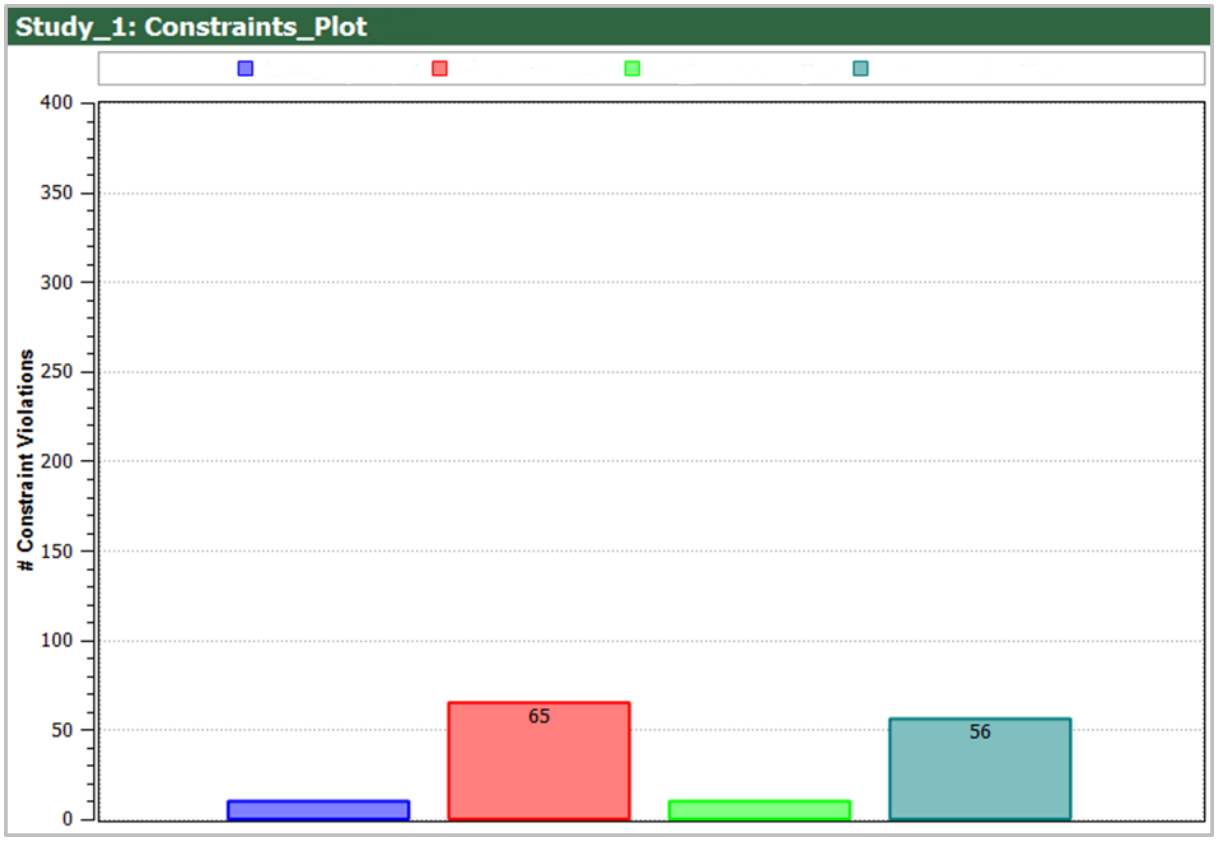

Figure 15. An example of a constraint plot for four constrains.

6. Case Study Using DAKOTA

The DAKOTA (Design Analysis Kit for Optimization and Terascale Applications) is a tool, which provides iterative analysis methods and interface to work with simulation codes. Software is capable of doing scalar and multi-objective optimisation, and uncertainty quantifications, DOE studies, parameter estimation, and sensitivity and variance analysis. DAKOTA is developed at Sandia National Laboratories New Mexico and the development project has started in 1994. The DAKOTA is an open source tool and the version 5.3.1, which was used in the case study, is licensed under the GNU Lesser General Public License (LGPL).

A schematic presentation of an iterative DAKOTA process is shown in Figure 16. A study in DAKOTA is defined as a textual DAKOTA input file (.in extension), which defines design variables, and responses for the study, as well as the study type, parameters and other options. For each evaluated design DAKOTA writes a parameter file, which is used to create the input file for the simulation execution. The simulation process is executed as an external process using a simulation script. Responses for the design evaluation are gathered into the DAKOTA result file from the simulation output file. The coupling between DAKOTA and the simulation process is done using input and output files. This approach is called a black-box coupling, because DAKOTA doesn't need to know the internal functionality of the simulation code [5]. When the DAKOTA process is finished, variables and responses of evaluated designs are written into DAKOTA output file (.out extension).

6.1 Preparing the Simulation Model for the Optimisation Project

In this case study DAKOTA runs a simulation process of Adams (see Chapter 4). Simulation in Adams can be done from the graphical pre-processor tool Adams/View or from the command line by calling the Adams/Solver directly. With the HEEDS MDO case study the simula-

tion was executed by calling the Adams/View and running simulation commands with it, while in this DAKOTA case study the simulation is done by calling the Adams/Solver directly.

When using the Adams/Solver directly, the Adams simulation model needs to be described as an Adams/Solver dataset file, which is a complete textual description of the model using Adams/Solver Data Language statements [8]. The Adams/solver dataset file of a simulation model can be exported from Adams/View. An essential thing about the Adams/Solver dataset file is that the model structure is flattened. This means that for example design variables used in the model are replaced with the design variable values and submodel structures defined in the model are not available anymore. For the use of the Adams/Solver dataset file in a DAKOTA optimisation loop, the model flattening means that the parameterization done in the simulation model cannot be used anymore. To be able to use Adams/Solver dataset files in a DAKOTA loop, all instances of needed design variables in the dataset file has to be tagged manually. The easiest way to do the tagging is to replace the value of the design variable in the model with a unique value, then exporting the model as an Adams/Solver dataset file, and then replacing the unique value strings with tagged variable names. In this case study design variables are tagged using variable name in <> quotation marks.

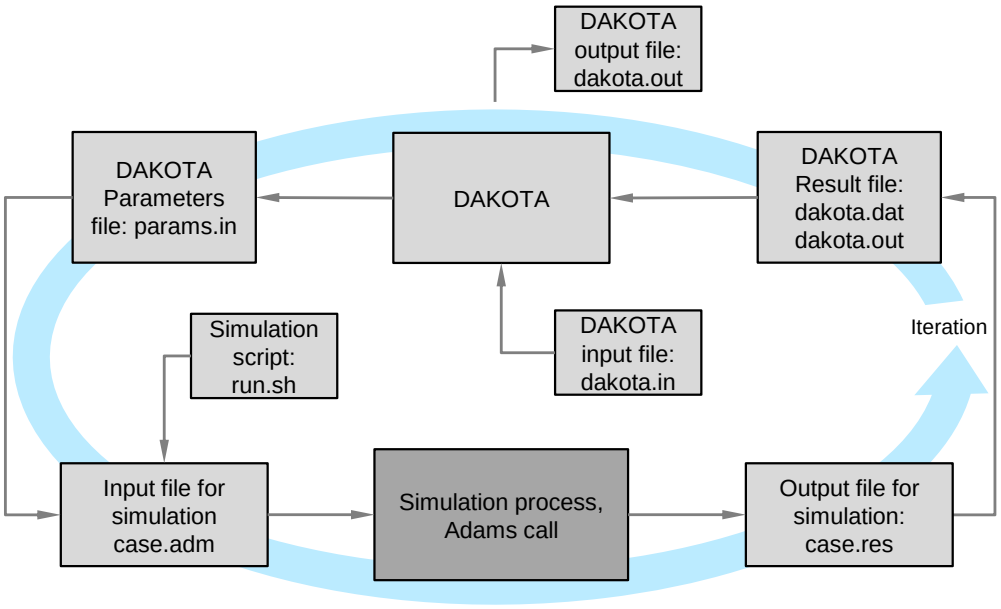

Figure 16. A schematic picture of the DAKOTA iterative process with a simulation code [5].

Despite the need for the manual tagging of design variables, the use of the Adams solver dataset files is easy and efficient, because the format of the Adams/Solver dataset file is clear and using string replacement the tagging is straightforward to do. The limitation to the use of the dataset file is that geometries in the simulation model cannot be parameterised. During the model flattening mass and inertia properties of all bodies based on the geometry are calculated and after this dimensions of geometry definitions are not available in the dataset file anymore.

Before the tagged Adams/Solver dataset can be sent to the Adams/Solver in a DAKOTA process, the tags have to be replaced with appropriate values of the design variables. This is done using a Python pre-processing script, which does the string replacement for the tagged variables. After that the simulation can be executed in shell script (in Linux environment) using a command

```
aview2013.1/mdi -c ru-st i run.acf
```

which tells that Adams/Solver is called from command line (*-c*) as a standard version (*ru-st*) in interactive mode (*i*) using the solver command file (*run.acf*) [8] (note that in Linux environment the command line option *-c* is needed, while in Windows environment it is not). The solver command file *run.acf* contains the following lines

```
template.adm
case
sim/sta
sim/dyn, end=25.0 dtout=0.005
```

It defines the name of the Adams/Solver dataset file (*template.adm*), name of the output files (*case.res*, *case.msg*), and simulation commands. In this case there is the calculation of the static equilibrium (*sim/sta*) and after that dynamic analysis with duration of 25.0 s and output time step of 0.005 s (*sim/dyn, end=25.0 dtout=0.005*).

6.2 Defining the Study

DAKOTA process is defined using a DAKOTA input file. An example of a DAKOTA input file (d*akota_example.in*) for parameter optimisation with two design variables, one objective function and one inequality constraint function is shown in Figure 17. The structure of the input file is divided into six sections using keywords strategy, method, model, variables, interface, and responses. The strategy section acts a control layer for the DAKOTA process. In this case *single_method* is used, which means that one model is run in one iterative process. Other available strategies such as hybrid optimisation or Pareto optimisation are used in DAKOTA's advanced optimisation approaches [5]. Strategy section includes also the definition of DAKOTA's tabular output file using keyword *tabular_graphics_data*.

The method section defines the type of the DAKOTA study, which can be parameter study, optimisation, uncertainty quantification etc. In this case an optimisation method called *coliny_direct* is used. Each method type contains own set of parameters. For the used optimisation method parameters *max_iterations* and *max_function_evaluations* are defined.

The model section defines the interface between the DAKOTA process and the simulation process. The keyword *single*, which is also the default case, defines that design variables and responses of a DAKOTA process are connected using a single interface through a direct simulation evaluation. More advanced options under the model section are used to define multi-level processes with sub-models and sub-iterators.

Variables section defines the design variables of the study. Design variables can be continuous or discrete. In the example there are two continuous variables with lower and upper bounds, and descriptors, which define the tagging names used in the Adams/Solver dataset file.

Under the interface section the connection between the DAKOTA and the simulation code is defined. To run an external simulation code as a separate process either a *system* call or a *fork* call can be used, while *fork* is strongly recommended by the manual, especially when multiple parallel analyses are run asynchronously. The difference between *system* and *fork* calls lie in the system functions that are used to run the process. In both calls the coupling between DAKOTA and the simulation process is done using input and output files. In large DAKOTA runs the amount of this kind of file I/O can be a bottle neck. For such cases the third interface option *direct* can be useful. In this option the simulation code needs to be converted to a subroutine and linked into the DAKOTA executable. The use of this *direct* interface makes the process managing simpler and decreases the file I/O. Direct function inter-

faces are available for Matlab and for instance Sandia's structural dynamics code Salinas and multiphysics framework SIERRA [5].

The analysis driver defines the name of the test function, in this case *run.sh*, which includes commands to be run during each design evaluation. The first command in the shell script *run.sh* is Python pre-processing script, which reads the design variables values from the DAKOTA's parameters file called *params.in* and replaces the tagged variables in the Adams/Solver dataset file with these values. The second command calls Adams/Solver to run the simulation. The third command is Python post-processing script, which reads the result file of the simulation, calculates response values, and writes them into DAKOTA's results file called *result.out*.

All input files and also output files (which may be empty) that are involved in the simulation run are located in a template directory called *case_template*. Each simulation run is invoked in an own directory. This functionality is defined using the keyword *work_directory*. When a new design evaluation is started, DAKOTA creates a new work directory called *case*, copies all files and directories defined in the template directory into it using soft links and runs the simulation. Input and output files and directories defined at the *case_template* directory are the following:

-	*template.adm*	Input	The Adams/Solver dataset file
-	*run.acf*	Input	Solver commands
-	*run.sh*	Input	The shell script for the DAKOTA loop
-	*case.res*	Output	Result file from the case evaluation
-	*case.msg*	Output	The message file from the case evaluation
-	*pyPreProcess.py*	Input	Python pre-processing script
-	*pyPostProcess.py*	Input	Python post-processing script
-	*TyreData*	Input	Directory containing tyre data files
-	*RoadData*	Input	Directory containing road data files

For an evaluated case the case directory contains also two DAKOTA files:

-	*params.in*	Input	DAKOTA parameter file
-	*results.out*	Output	DAKOTA results file

The interface section includes also the definition of parallelization during the process evaluation. DAKOTA provides large variety of mechanisms for the parallelization from a one multiprocessor workstation to a computing cluster. The utilization of parallelism depends on the study type and algorithm. Global and derivative free optimisation methods available in DAKOTA are capable for parallel computing by dividing the search area into sub-problems that can be sent to separate computing resource. In the example case parallel computing is defined using keyword *evaluation_concurrency* and two evaluations can be run at the same time. The evaluation is done in an *asynchronous* mode, which, means that parallel evaluations are not synchronised and both computing resources may start a new evaluation when the previous evaluation is finished.

Response section contains the definition of response functions of the DAKOTA process. In the example optimisation problem there is one objective function and one nonlinear inequality constraint function with upper bound defined. Response function values are found from the DAKOTA results output file using the descriptors names. In a case of multiple objective func-

tions the weighting of objective function values is available. Response section also includes the definition of response function's derivative (gradients) and second order derivative (hessians) usage. For the example case of derivative free optimisation problem gradient and hessian information is not available.

DAKOTA installation package includes a wide collection of complete DAKOTA example cases. When a new DAKOTA process is to be defined, a good starting point is to take a suitable DAKOTA input file from the example cases, and modify it using a text editor. A useful tool, when modifying or defining a DAKOTA input file from the scratch, is also JAGUAR from Sandia. It is a graphical tool to generate the DAKOTA input file specifications by suggesting available keywords for input file sections. An example view of JAGUAR graphical user interface is shown in Figure 18. Features of JAGUAR to help the input file definition are syntax checking, highlighting, auto-completion of keywords, text formatting functions, and links to the DAKOTA online help. [10]

```
# Dakota Input File: dakota_example.in
strategy
    graphics
    tabular_graphics_data
        tabular_graphics_file = 'dakota_example.dat'
    single_method

method
    max_iterations = 500
    max_function_evaluations = 1000
    coliny_direct

model
    single

variables
    continuous_design = 2
        lower_bounds        10.0                 10.0
        upper_bounds        100.0                200.0
        descriptors         'spring_stiffness'   'spring_damping'

interface
    analysis_drivers 'run.sh'
        fork
            parameters_file 'params.in'
            results_file 'results.out'
            file_save
            work_directory
                named 'case'
                directory_tag
                directory_save
                template_directory 'case_template'
    asynchronous
        evaluation_concurrency 2

responses
    descriptors 'Obj_function' 'Constraint'
    objective_functions = 1
    nonlinear_inequality_constraints = 1
        upper_bounds        0.001
    no_gradients
    no_hessians
```

Figure 17. An example of DAKOTA input file for the parameter optimisation case.

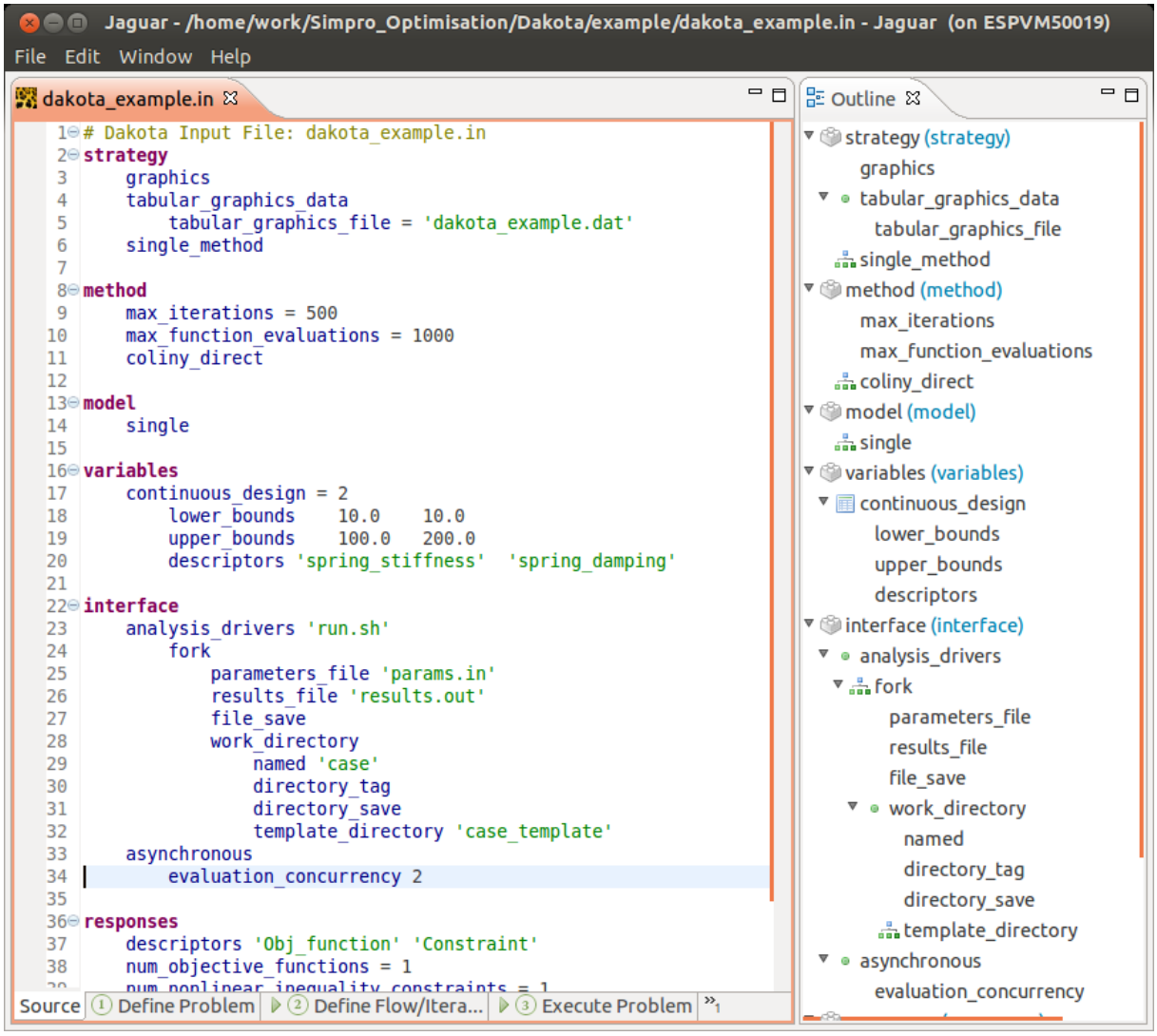

Figure 18. View of the JAGUAR GUI.

6.3 Running the Study

The parameter optimisation case study using DAKOTA is run in Linux environment. The input file of the process is basically the same as the example case shown in the previous chapter except the number of design parameters, objective and constraint functions and their boundary values. In this case study the weighted sum of objectives functions with inequality constraint functions is optimised.

The optimisation method used in the example case is called Division of RECTangles (DIRECT), which is a global derivative free method. DIRECT algorithm divides the design space into subspaces and focuses the search on the promising areas. There are two implementations of the DIRECT algorithm. The used version is *coliny_direct*, because it supports nonlinear constraints, which are defined in the case study. In the case study necessary parameters needed for the *coliny_direct* are the maximum number of iterations and function evaluations. The maximum evaluation concurrency for this method is twice the amount of design parameters. An alternative global derivative free method for the case study would be an evolutionary algorithm. It simulates the evolutionary process by defining design evaluation generations, where the fittest ones of generation can reproduce and form new design evaluations and finally converging to the optimum design.

Before running the DAKOTA input file, it can be checked using command

```
dakota –i dakota_example.in –check
```

DAKOTA run using specified input and output files can be started using a command

```
dakota –i dakota_example.in –o dakota_example.out
```

DAKOTA can also be run in additional execution modes: (1) pre-run, which generates the set of points at which the system will be evaluated, and (2) post-run, which defined final statistics of a run. DAKOTA has also restart functionality. Using .rst file a DAKOTA run can be continued, if the execution has stopped for some reason. All DAKOTA run modes can also be executed from the JAGUAR.

6.4 Viewing the Results

The optimisation run with maximum of 1000 iterations was performed. DAKOTA finished the run after evaluation number 507. The objective function value as a function of design evaluation number is shown in Figure 19. It can be seen the rate of the change for the performance value is high for the first about 80 designs and after that the gain is small. The amount of objective function change after the 80 design is small and smaller amount of designs would have been sufficient. Normalised objective function values as a function of design evaluation number is shown in Figure 20.

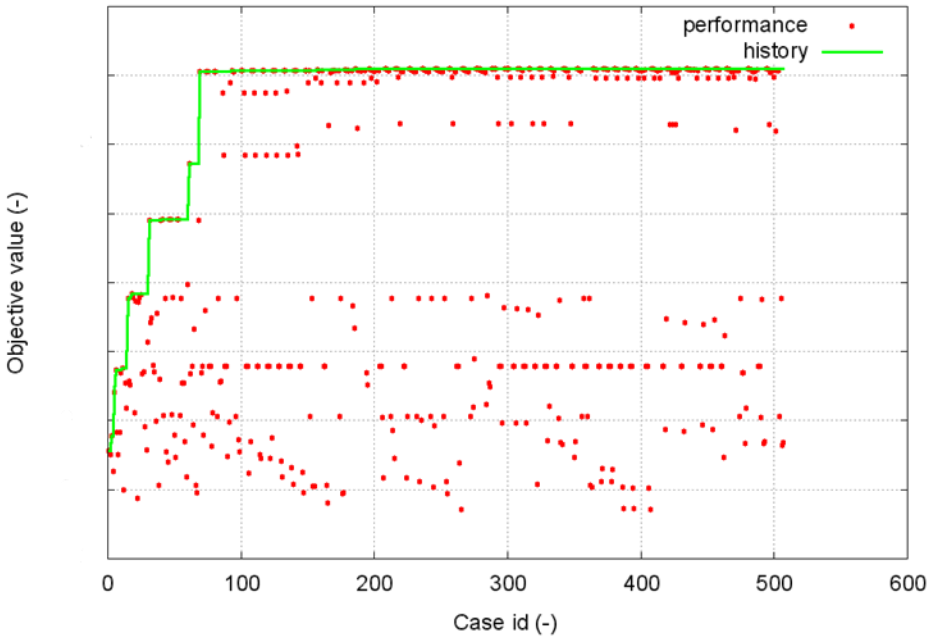

Figure 19. Objective function value as a function of design id.

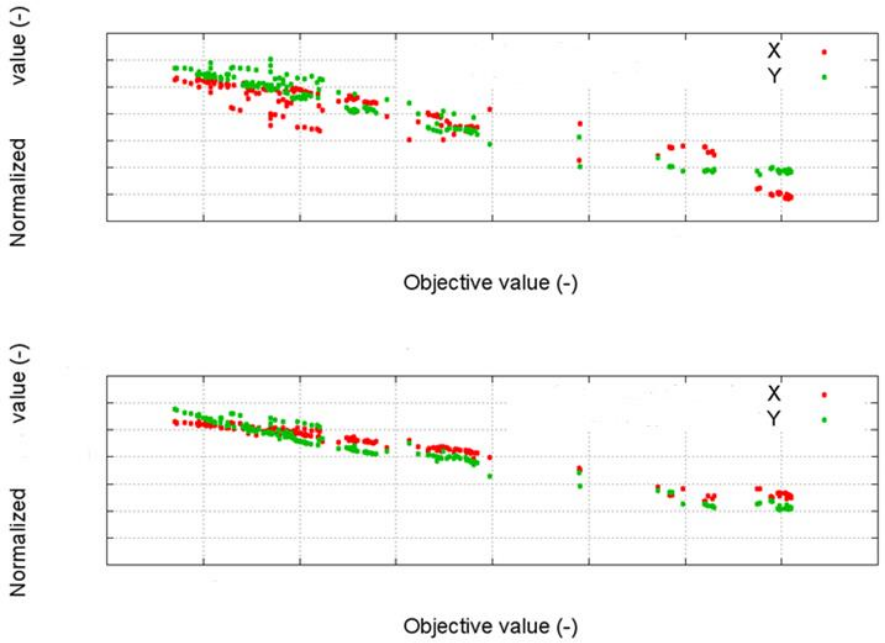

Figure 20. Normalised individual objective values as function of objective function value.

7. Conclusions

7.1 Case Studies

The case studies for the parameter optimisation applied to a multibody simulation were successful. Two different case studies were defined and run with two different optimisation tools. Both case studies were run with the default parameter values of the optimisation methods. Optimisation tools and the mathematical algorithms inside them were found capable and robust for parameter optimisation of multibody models without extensive knowledge of optimisation before.

The first case study was done using HEEDS MDO. The definition of an optimisation processes using the graphical user interface was found easy and straightforward. Clearly formatted user interface with active helping instructions makes the definition of analyses and connections efficient. Tools for tagging the general input and output file are fluent and easy due the visualization of the file. As a new feature HEEDS MDO version that was used in the case study had an interface to connect an Adams/View simulation model to the process. The use of this so-called “Adams portal” was found user-friendly and efficient. Using the portal the Adams binary file is imported and design variables and design objectives defined in the Adams model can be tagged and used directly in HEEDS MDO process. For the optimisation process solving the default optimisation method SHERPA was found to be robust as all tested cases were successful. Visualization of results could be done quickly using plotting functions within HEEDS MDO. The further post-processing and visualization of results was done in HEEDS POST. Disadvantage of post-processing tools was the modification capabilities of plots. For example axis definitions or labels font characteristics were found difficult or impossible to define. This would be needed to get clear and understandable results plots in some cases.

The second case study was done using DAKOTA code. Starting from the example cases that come with the DAKOTA installation, the definition of the optimisation process was straightforward. When new features are needed to be implemented to the DAKOTA input file, JAG-

UAR can ease the definition remarkably. As a command line tool for optimisation analyses DAKOTA needs more attention in the beginning but after a while it is efficient to use. The usage of DAKOTA needs some scripting because interfaces between the simulation and post-processing codes and DAKOTA need to be defined. The value of DAKOTA is that it is a capable general tool for any process and it has open source licensing.

7.2 Optimisation Process

Even though the definition and execution of the optimisation process can be easy and straightforward, some issues are important to be checked to get a successful optimisation study. First of all, it is important to understand the case that is being optimised. This means that the dynamics of the system and the effects of design parameters on the system responses should be at least roughly known. Valuable information can be get for example by performing DOE studies for the studied design parameters. Then regarding to the studied system, all relevant excitations of the system has to be present. Otherwise the optimum solution may be relevant in the mathematical sense, but has no relevance in the real system design.

The main components of the optimisation process are design parameters, objective and constraint functions and the solving method. Case studies were defined with four objectives, which were different in their nature. The sum of objective values was defined and thus the objective function was as a scalar. If objectives values would have been competing against each other's, multi-objective study had to be declared. Results of a multi-objective study are given as Pareto plots, which show the equal designs. In multi-objective study one optimal solution is not given.

When multiple objectives are treated as a sum, an important aspect is the normalization and weighting of objective functions. The normalization is used to scale each objective value to the same range to get them effecting on the weighted sum with the same proportions. Otherwise the largest objective value can dominate the objective function value while the smaller magnitude value can have hardly any effect at all even though it may be significant for the optimised design. In the studied cases the objective values of the baseline design were used as the scaling factors. This is needed to take into account the different scale of different objective values. The weighting factor defines the importance of an individual objective value on the scalar objective function value. An example of the weighting could be the optimisation of total material costs. Prices of different materials can be used as the weights when objective function components are the masses of different material on the design. In the studied case the weighting was not needed and weighting factors were defined as unity.

During the iterative optimisation process design variables of the simulation needs to be updated and the simulation code called from the command line. Approaches to update the model and perform the simulation differed between the studied cases. HEEDS MDO's Adams portal called Adams/View, which updates the design variables and performs the simulation using Adams/View commands. In the DAKOTA case the simulation model was defined as a text file. The tagged design variables were updated using string replacement and the modified model file was sent to Adams/Solver. The capabilities of these approaches differ a lot. When design variables are updated inside Adams/View, it gives possibilities to wider model modifications during the model update using Adams/View command language. Design variables can be used for example for the geometry while in the flattened Adams/Solver input it can be used only in function expressions. Optimisation tools didn't limit the use of these approaches; both approaches could have been used with both tools.

The core of the optimisation process is the algorithm which manages the iterative process. In HEEDS MDO and DAKOTA there are several algorithms available, which are suitable for different type of systems. In a case of a multibody model, an optimisation of a non-linear system is needed. When using multibody models the gradient is not available, unless it is approximated somehow and thus gradient-free methods needs to be used. In the case studies

the focus was on the optimisation process and therefore different methods were not studied. After the optimisation process is defined and solved the process continues by defining the properties of the found optimum point. Valuable information can be the behavior of the system near the optimum point, which can be done using sensitivity analysis.

8. Summary

The use of optimisation gives a way to find optimal designs systematically. This is needed in engineering work, when design aspects are competing against each other and the making of design choices often lead to the making of compromises. In parameter optimisation the system performance, which is defined as a number or a set of numbers, is optimised by varying design variables of the system inside a defined range.

In this work the parameter optimisation problem definition and typical methods for solving were studied in general. Two parameter optimisation case studies were performed using a multibody simulation model of an elevator. The selected vibration feature was optimised using several design variables of the system. The feature was evaluated using a scalar valued objective function and also inequality constraints for the optimisation were defined.

Case studies were done using software HEEDS MDO and DAKOTA. Both case studies were run with the default parameter values of the optimisation methods. This means that the tools and especially the mathematical algorithms inside them are capable and robust for parameter optimisation of multibody models. In the studied cases both tools could be used successfully even by an unexperienced optimisation user.

References

[1] Eberhard, P., Bestle, D., Schiehlen, W. *Optimization of Mechanical Systems.* In a publication: *Advanced Design of Mechanical Systems: From Analysis to Optimization*. Udine: CISM Courses and Lectures, 2009. Vol. 511, p. 237−252. ISBN 978-3-211-99460-3.

[2] Lodewijk, F. P. E. Optimization of Multibody Systems using Approximation Concepts. Eindhoven: Technische Universiteit Eindhoven, 1997. 140 p. ISBN 90-386-0520-X.

[3] Fu, M. C., Glover, F. W., April, J. *Simulation optimization: A Review, new developments, and applications*. Proceedings of the 2005 Winter Simulation Conference. 14 p.

[4] Tekin, E., Sabuncuoglu, I. *Simulation optimization: A comprehensive review on theory and applications*. IIE Transactions, 2004. 30:11, p. 1067−1081. ISSN 0740-817X.

[5] Simbierowicz, G., Kortelainen, J. *Assessment of different computational methods used for estimating the lateral quaking in a high-rise elevator*. Proceedings of the 20th International Congress on Sound and Vibration, 2013. 8 p. Available at: http://www.icsv20.org/content/papers/papers/full_paper_185_20130411190404499.pdf

[6] *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis. Version 5.4 User's Manual*. Livermore: Sandia National Laboratories, 2009. 333 p. Available at: http://dakota.sandia.gov/docs/dakota/stable/Users-stable.pdf. Cited 11.2.2014.

[7] *HEEDS MDO, Multidisciplinary design optimization software*. East Lansing, Michigan, USA: Red Cedar Technology, 2013. 4 p. Available at: http://www.redcedartech.com/pdfs/HEEDS_MDO_Brochure_web.pdf. Cited 11.2.2014.

[8] *Adams Online Help. Version 2013.1*.

[9] *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis. Version 5.4 Reference Manual.* Livermore: Sandia National Laboratories, 2009. 237 p. Available at: http://dakota.sandia.gov/docs/dakota/5.4/html-ref/index.html. Cited 11.2.2014.

[10] Adams, B. M., Chan, E., Lefantzi, S., Ruthruff, J. *DAKOTA JAGUAR 2.1 User's Manual*. Livermore: Sandia National Laboratories, 2011. 41 p. Available at: http://dakota.sandia.gov/docs/dakota-jaguar/2.1/JaguarUsersManual2.1.pdf. Cited 13.2.2014.