

Investigation and Application of Multi-Disciplinary Optimization for Automotive Body-in-White Development

Allen Sheldon*, Edward Helwig*, Yong-Bae Cho**

* Honda R&D Americas, Inc.

** CSM Software, USA

Abstract:

A process has been created for applying multi-disciplinary optimization (MDO) during the development of an automotive body-in-white (BIW) structure. The initial phase evaluated the performance of several different optimization algorithms when applied to structural MDO problems. From this testing, two algorithms were chosen for further study, one of these being sequential metamodeling with domain reduction (SRSM) found within LS-OPT.

To use the LS-OPT optimization software effectively within a production environment, adaptations were made to integrate it into an established CAE infrastructure. This involved developing a LS-OPT server and architecture for the parallel job submission and queuing required in the MDO process. This enabled LS-OPT to act as an integral part of the enterprise CAE architecture as opposed to a standalone tool.

Within this integrated environment, the SRSM method has been applied to an MDO process that combines 7 load cases and takes into account crash and NVH requirements. The objective of the MDO was to minimize mass while constraints enforced the performance requirements of each load case. The thicknesses of 35 parts were considered in this MDO. The application of the SRSM MDO strategy resulted in an optimized design with a 6% weight reduction for the portion of the BIW considered. The optimized design was determined with reasonable computational resources and time considering the computational intensity of the analysis.

1 Introduction

As automotive body-in-white (BIW) development times continuously shrink, traditional CAE analysis used to develop the body structure can become multiple processes that must run in parallel as separate disciplines. If the interaction between these disciplines during development is minimal the result might be an overlap of design improvements that could result in an unnecessary mass increase of the BIW structure. As shown in figure 1, by placing a multi-disciplinary design optimization (MDO) within the development cycle, not only can a more efficient BIW structure be developed, but a mass reduction otherwise unobtainable by traditional means may be achieved.

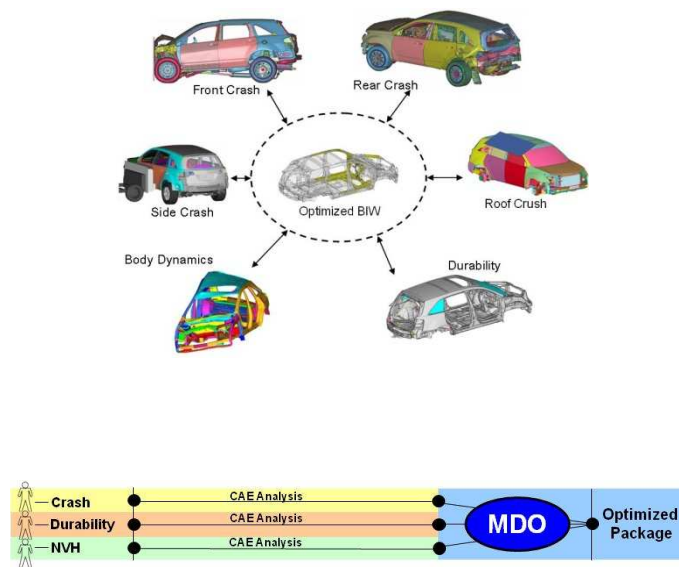


Fig. 1. An example of a MDO process and its application within a development cycle.

To develop the MDO process, first three test problems were optimized using a number of different MDO techniques. This testing identified a few techniques that showed promise for further use. One technique, which showed good performance for a single objective MDO, was the sequential response surface technique with domain reduction (SRSM).

2 Assessment of MDO Methodologies

When faced with the task to optimize a structure such as a BIW, consideration must be given to which optimization algorithm would be likely to perform best. If the structure's response is likely to be linear, or if only a very small improvement is desired, perhaps a local optimization method based on gradient information will suffice. But if the response is likely to be highly nonlinear and a global optimum is sought, then it is unlikely that a local optimization method based on gradients would suffice. Instead, methods that can find a global optimum in the face of highly nonlinear responses would be required.

These methods could range from an indirect method using response surface techniques to a direct heuristic method based in nature such as a genetic algorithm (GA) as shown in figure 2.

The challenging part from an analyst perspective is that the choice of a method is largely dependant on knowing the type of issues you are dealing with (e.g., linear or nonlinear, continuous or non-continuous, unimodal or multimodal, and others), but often that information is not available until an optimization has been executed. This can present a challenging dilemma: how to choose the best optimization method before knowing the exact nature of the issues. In the case of applying a MDO process to BIW developments, test problems were developed that represent a reduced, but typical application. By studying the performance of a group of algorithms on these test problems, each could be evaluated and those algorithms with merit could then be used in further development of the MDO process.

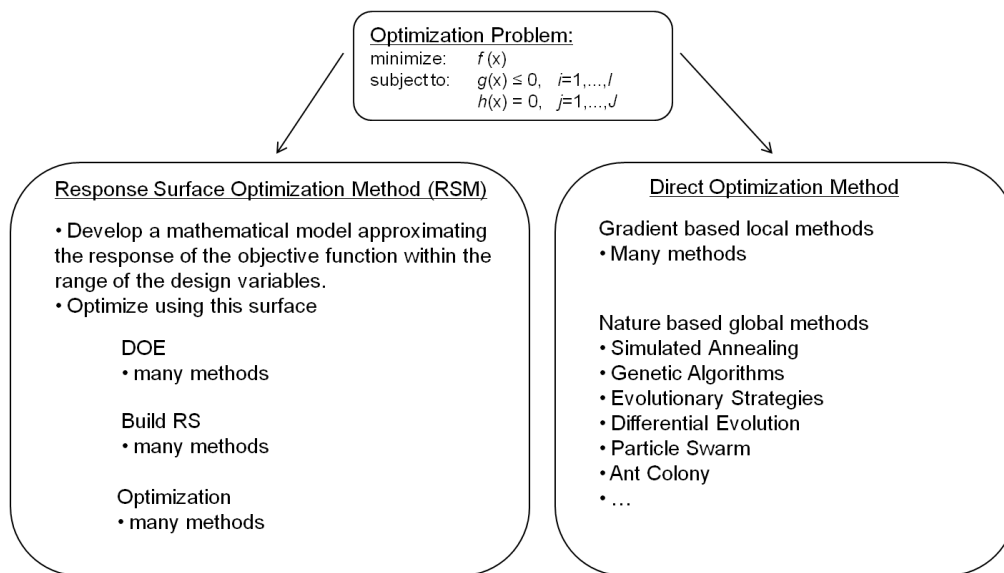


Fig. 2. Some of the methods available for structural MDO application.

2.1 Test Problems

The test problems considered a reduced set of disciplines to keep the runtime for the MDO within reason. The models used in these test problems were also smaller in terms of element count than current models in use for BIW development. But, the models still captured the challenging nonlinear, unstable, and non-continuous nature found when applying optimization to crash and NVH models. Three test problems were used and are shown in figure 3.

The first problem was a thickness optimization with performance constraints from body stiffness, side impact, and roof crush analyzes. The thicknesses of 14 parts were considered as continuous variables and the objective was to minimize the mass. To increase the difficulty for the optimization algorithm, the simulations were initialized to start with the maximum value for each part's thickness. Also, the requirements were increased by 20%, but with the parameters initialized to the maximum values the optimization did start from a feasible design point.

The second problem was similar to the first, but material variables were added to the thickness variables. The materials were discrete variables chosen between a set of material IDs. A cost function was defined for the different materials. This was a multi-objective optimization where the mass and cost were both to be minimized and a Pareto front identified.

The third problem was directed toward shape optimizing using mesh morphing in ANSA, or parametric CAD geometry from CATIA. The purpose was to understand the setup and automation of the interaction between the optimization process and the mesh morphing or CAD software.











| | | Test Problems | | |
|---------------------|---------------|---|--|---|
| | | MDO | MDO + MOO | MDO + Shape |
| Modes / Constraints | Crash | Side Impact Roof Crush  | Side Impact Roof Crush  | Side Impact Roof Crush  |
| | NV | Global Modes  | Global Modes  | Global Modes  |
| | Durability | | | |
| Variables | Thickness | 18 Parts  | 7 Parts  | 3 Parts  |
| | Material | | 7 Parts  | |
| | Shape | | | 1 Shape Variables |
| Objectives | Minimize Mass | ● | ● | ● |
| | Minimize Cost | | ● | |

Fig. 3. Test problems used to evaluate MDO methods.

2.2 Methods Evaluated

Various commercially available optimization software codes were chosen for this evaluation. A recommendation was given by the vendor of each code as to the best method to try for each of the test problems. This resulted in a number of different methods being evaluated giving a good representation of what is currently available for optimization. The methods evaluated included a multi-start gradient based method, differential evolution (DE), genetic algorithm (GA), evolutionary strategy (ES), response surface modeling (RSM), and some methods which use a combination of these and other method for its global search. Not all methods were tried on each problem—only those thought by the vendors to hold a promise for producing good results.

Of course, one evaluation criteria to evaluate the optimizer’s performance was to look at the results obtained for the optimum design found by each optimizer. But, also of great importance for evaluating

the performance was the time required to find that optimum design and the efficient use of the computing resources. All methods were allowed to run a very similar number of design evaluations (slight differences in population numbers between methods prevent one from using the exact same total number of jobs) so that the overall run times could be compared. Other more subjective criteria included the ease of use of the software for set-up, interfacing to other analysis software, pre- and post-processing, restarting an optimization, and software cost.

2.3 Results

2.3.1 Single objective MDO test problem

The results for the MDO test problem are shown in figure 4. The baseline results are shown for reference. This baseline result represents the best design obtained through manual iterations that achieved all of the performance requirements. For this evaluation, it was assumed that no optimized design would have a lower mass than the baseline since the performance requirements were increased 20% above the baseline values.

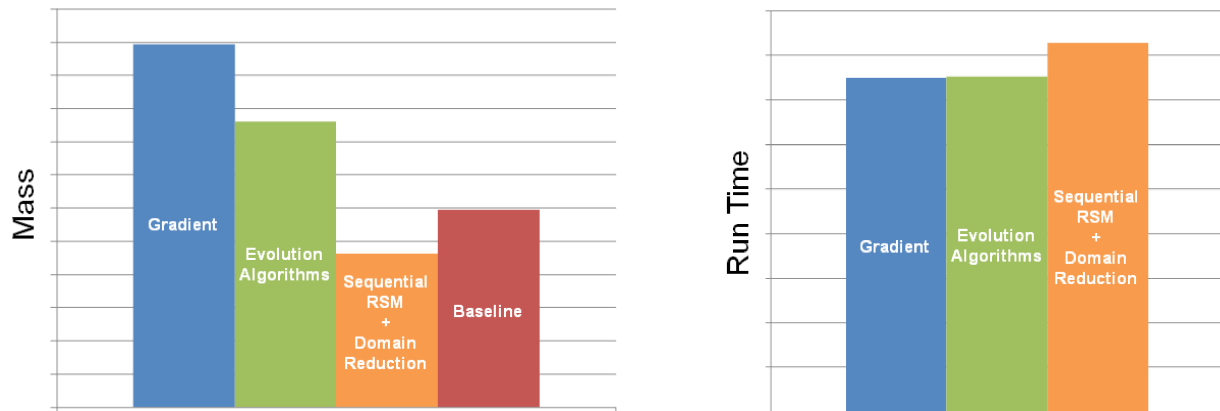


Fig. 4. Results from the single objective MDO test problem.

The gradient method (even with its multi-start character) yielded the optimized design with the lowest mass reduction. Of course, this relatively small mass reduction is assumed to be a result of the method's local optimization nature and the likely possibility, with a problem containing highly nonlinear results, that it got trapped in local optima. Due to its early convergence at a local optimum, the gradient method's total run time was the shortest. But it had the poorest efficiency for use of the compute resource due to its nature: it could run parallel jobs when computing directional sensitivities, but during descents it was limited to only a few jobs. This leaves much of a dedicated computing resource idle at times during the optimization.

The next methods tested consisted of a number of the evolutionary algorithms. Those tested included a GA, hybrid GA (combined a GA with other methods to assist its search), ES, and DE. These methods are shown together in figure 4 since the results were very similar. Slight differences in the run time occur depending on the population size required, but as long as one matches the population size to the computing resource available, the resource will be used efficiently with each method. The ES and hybrid

GA methods did have a potential advantage of being able to run much smaller population sizes. This could allow the method to make faster progress toward an improved design if the only a small computing resource is available for the optimization.

The other method tested was a sequential response surface method with domain reduction referred to within LS-OPT as the SRSM method [1]. The SRSM method yielded the design with the lowest mass. Surprisingly, it found a design with a lower mass than the baseline model even though the performance requirements had been increased by 20%. The time was 9% longer than the evolutionary algorithms, but the tolerance for convergence was set to a change of less than 0.01 on both the design and objective functions. It appeared from the optimization history that if the convergence criteria had been applied to only the objective function change it would have stopped sooner with a very similar mass reduction.

2.3.2 Multi-objective MDO test problem

The results from the multi-objective optimization (MOO) evaluations are shown in figure 5. This problem consisted of compute-intensive simulations that must be run in parallel, multi-objectives, and discrete variables. Disappointingly, this combination reduced the software choices that could be evaluated since many did not support some combination of these features. The methods that were evaluated included a single stage RSM, a sequential RSM, and a multi-objective GA (MOGA). The two objectives for this problem were mass and cost. Because of the cost function used for the analysis, the optimizations all struggled to produce a Pareto or trade-off front. Instead the optimizations converged to a single optimum for the lowest cost and weight design. Thus, we do not compare the Pareto fronts, but instead look at the optimal mass and weight found.

Figure 5 shows that the optimizations all had similar results for the mass and the cost. The RSM-based methods found slightly better optimal values than the MOGA, but the biggest difference was seen in the run times. The RSM-based methods were able to find an optimum in a fewer number of evaluations than the MOGA methods. With this MOO problem, we found no clear advantage to using a sequential RSM method over a more typical single stage RSM as they both had similar results.

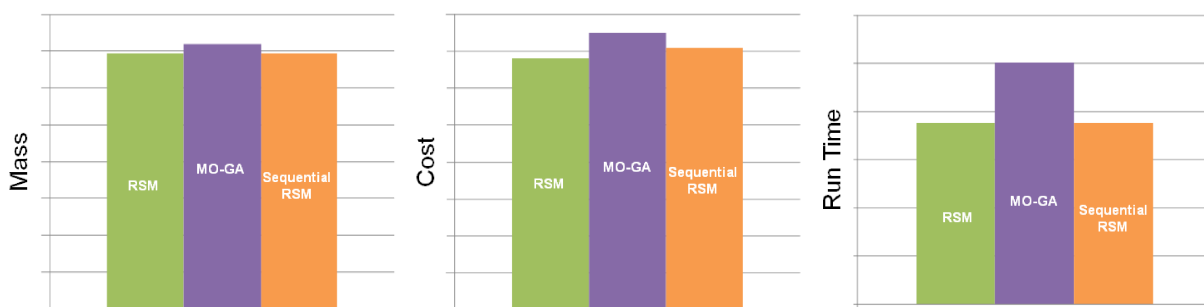


Fig. 5. Results from the multi-objective MDO test problem.

2.2.3 Shape optimization

The primary purpose of the third test problem was to investigate the interaction of optimization software with a mesh morphing and parametric CAD software. Some of the optimization software

packages have direct interfaces to mesh morphing or CAD software. This made the setup of this problem very straightforward. Since ANSA creates a design variable text file and parametric changes in CATIA can be driven from a macro text file, even without the direct interface, it is a fairly straightforward problem to setup. But this was a simple problem in terms of the small number of variables. As the complexity of the problem increases, having a direct interface would be very advantageous.

For this problem, we used a multi-start gradient method, a single stage RSM, a GA, and a sequential RSM. The results shown in figure 6 make it clear that even a simple shape optimization problem can yield mass savings. All of the optimizations were able to reduce mass. By changing the shape of parts to increase their roof crush load carrying capacity, the thickness the parts could then be reduced.

Once again the multi-start gradient-based method appeared to converge in a local optimum. The other methods were able to find improved mass reductions compared to the gradient method. With the small number of variables in this problem, the RSM methods were both very effective at finding optimal designs and doing so with a small number of runs. While the GA method found a similar mass reduction to the RSM methods, it took a much longer run time.

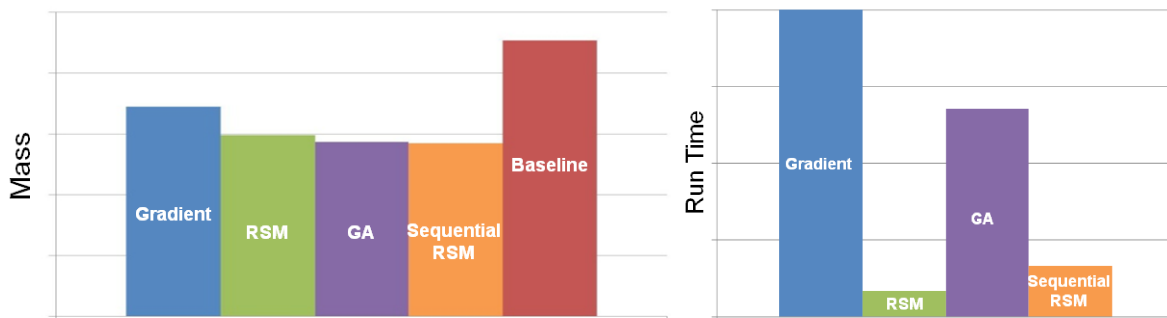


Fig. 6. Results from the shape optimization test problem.

2.4 Conclusions from MDO testing

Overall, the results from the test problems showed that the RSM methods performed well, even in the face of highly nonlinear responses such as those seen with automotive crash simulations. The evolutionary algorithms did not show a clear advantage compared to the RSM methods, but aspects of these methods remain attractive. Thus, we continue to work with these methods.

A question that remained was could a RSM method still work well with a more challenging problem whose complexity truly represented the MDO as it would be applied in a development process. This problem would consist of more load cases considered and more variables included than in the test problems. For the single objective MDO test problem, the SRSM method of LS-OPT yielded the result with the highest mass reduction. The remainder of the paper will focus on the integration of LS-OPT within an enterprise computing environment and the application of LS-OPT to larger multi-disciplinary optimizations of the body structure.

3 Integration of LS-OPT

A robust application to support advancements in optimization methodologies within our existing computing environment was needed. The goal was to develop a client and server architecture enhanced with a user defined interface for customization. The solution needed to be integrated into the existing high performance computing (HPC) environment, and designed to support hundreds of users including remote sites. The results of these collaborative efforts with the LS-OPT developers are included in the 4.1 release. The new enterprise environment solution using LS-OPT extends the capabilities beyond the previous stand alone desktop version of LS-OPT 3.x.

3.1 LS-OPT Server

The major changes to this new version of LS-OPT were in the internal scheduler system and the introduction of the LS-OPT Virtual Machine (LS-OPTVM). The LS-OPTVM can proxy job requests on behalf of the client, thus extending the standalone application to the enterprise environment. Figure 7 shows an overview of the LS-OPT server architecture.

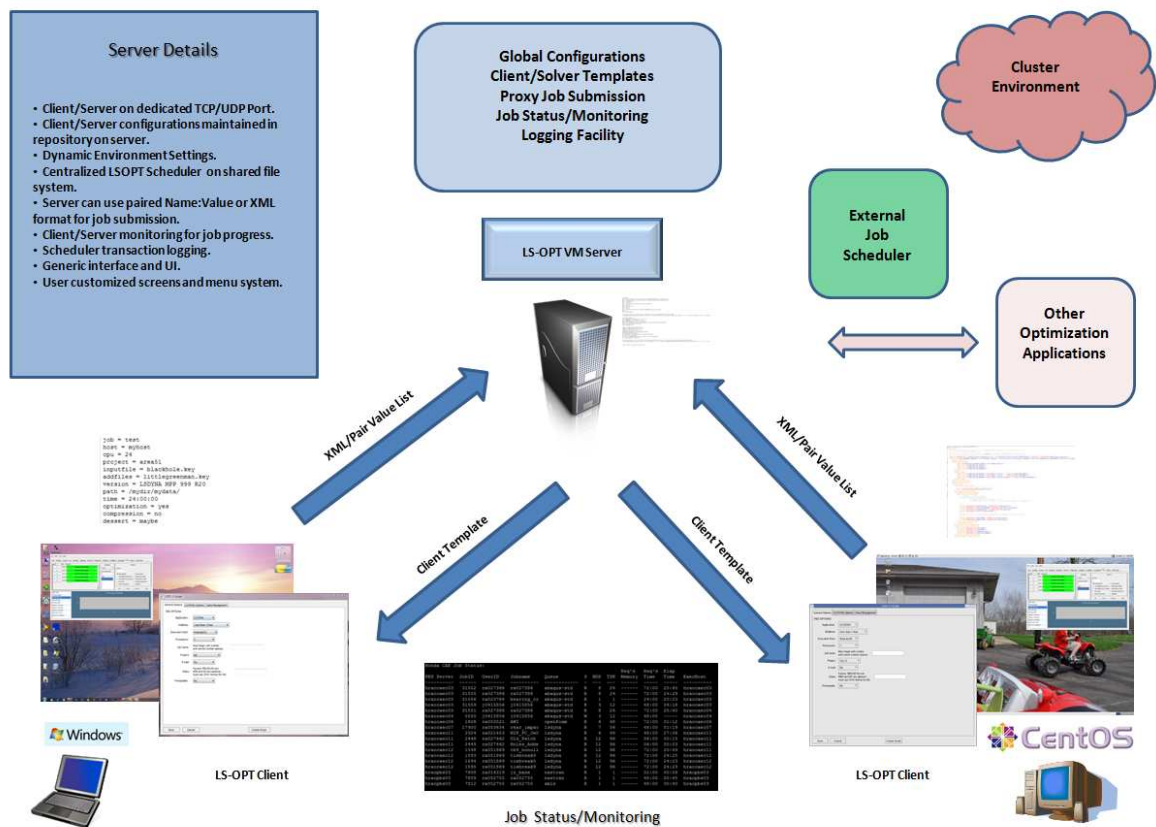


Fig. 7. LS-OPT 4.1 Server Architecture.

The internal job management system was changed to accept input from external sources for job monitoring. The commercial HPC Job Scheduler software was already producing standard output, which could now be used by the internal scheduler in the form of a formatted flat file. This information provided a mapping mechanism to compare the internal job ID to the external job scheduler. The logging facilities were enhanced to provide greater debugging capabilities necessary to understand the workflow process in the large HPC environment. The LS-OPT graphical user interface was changed to display the standard output from the HPC active jobs. As shown in figure 8, the job status bar now reflects the job progress and names of the user's optimization jobs. This change simplified identifying the user, active jobs, and the cluster resources.

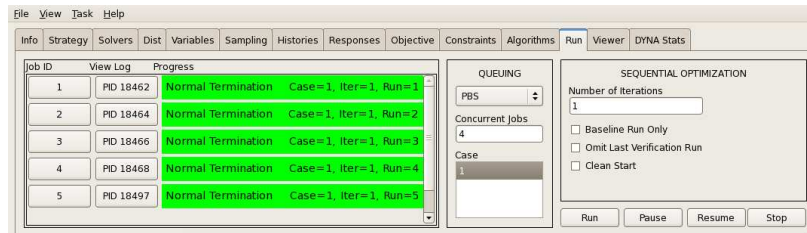


Fig. 8. The LS-OPT 4.1 job status interface.

All of these new features were added after the introduction of the unique “name=value” paired concept. These command line arguments values populated in the Env Vars window of the Solvers tab are the basis for job submission in the HPC environment. This mechanism allowed the LS-OPT application to directly interface into the HPC environment as an enterprise solution. The LS-OPT graphical environment can now be customized to match our internal job submission graphical interface with the use of the CScript language. This simple, yet robust language, works with wxWidgets to create the GUI configurations that can be customized by the user for supporting the diverse HPC environment requirements.

3.2 Architecture for Parallel Job Submission and Queuing

The LS-OPT 4.x application is configured to run from a network installation with support for both the Linux and Windows platforms. All configuration files and customization script are centrally located on redundant servers. This setup allows for ease of maintenance and configuration changes that can be immediately propagated to the clients and remote sites. The execution of the application invokes a TCP connection between the server and client. The client makes a request to the server for the graphical template corresponding to the solver to be used. The solver returns an XML-based file that configures the CScript to display the graphical interface necessary to populate the Env Vars tab. Figure 9 shows an example of the GUI for LS-DYNA. This GUI is needed to provide the arguments for the wrapper necessary to submit the job in the HPC environment. To submit the job, the client returns an XML file with arguments to the LS-OPTVM, which then submits with a wrapper or proxies the job submission on behalf of the client.

The user defines the number of jobs and iterations for the optimization study within the LS-OPT graphical interface. The LS-OPT internal scheduler will submit the entire sequence to the HPC scheduler to run on the production clusters. The jobs are run in parallel with predefined resources, queues and quotas. The jobs associated with the optimization study will be monitored once a minute by the LS-OPT

internal scheduler from the information provided by the HPC scheduler. The user will interface with LS-OPT to visually inspect the progress of the study. The user does not need to leave the LS-OPT interface because the environments are now fully integrated.



Fig. 9. CScript GUI for LS-DYNA job submission.

3.3 Results

Production testing for our methodology was completed with 5000 cores on HPC hardware using Intel and AMD processors. The data storage was NFS mounted volumes to commercial-based enterprise systems using 10GigE adapters. The networking was an Infini band infrastructure with Voltaire network switches to the storage nodes and clusters on the same 10GigE fabric. The external scheduling system was Torque 2.4.10. Approximately 76,000 jobs were run during a 15-day testing period. Less than one percent of the jobs failed due to software-related problems from LS-OPT or the external scheduler. Hardware failures could be mitigated with proper administration and monitoring tools such as Ganglia. Scheduling efficiencies could be realized by organizing shorter run times in between longer running iterations. The only limitation to the testing was the available resources. LS-OPT and the established processes scaled very efficiently in our environment.

4 Application of MDO for production

Here we describe the application of the SRSM optimization strategy to a large single objective MDO of a BIW structure. By increasing the load cases considered and the number of variables a production application was obtained.

4.1 *Setup*

This objective of the MDO was to minimize the mass through gauge optimization. To define the performance envelope for the body structure, seven load cases were used as follows: NV body stiffness modes, US NCAP 35mph flat wall frontal Impact, IIHS 40mph 40% front offset, FMVSS 301 50mph 30% offset rear impact, FMVSS 214 75 degree 20mph side pole, IIHS 90 degree 31 mph side impact, and FMVSS 216 roof crush. Constraints were applied for each load case to ensure that all performance requirements were met. The constraints were normalized to avoid weighting effects on large valued constraints. As shown in figure 8, the thicknesses of 63 parts were included in the optimization. This produced 35 design variables due to symmetry for many of these parts. All the variable thicknesses were set to vary continuously in a range appropriate for each part. These parts were defined in a single separate include file for analyses, and the variables were parameterized.

Upon an initial run of the optimization, it was quickly identified that the constraints used were not sufficient to help ensure a usable design could be clearly identified by the optimization process. This was primarily seen as unacceptable deformation in the floor area of the vehicle. To aid the optimization process in identifying truly acceptable designs, additional constraints were added that placed limits on the internal energy of key parts.

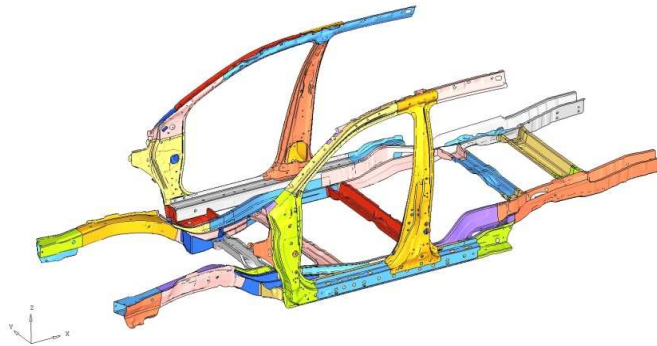


Fig. 8. Thickness variables for the parts shown were included in the optimization.

4.2 *Methodology*

The sequential response surface method with domain reduction was employed as a metamodel-based optimization strategy. For the metamodel, a radial basis function network was chosen for all load cases, but Linear D-optimal was used for the first iteration. Global sensitivities were also computed. Initially, 10 iterations were defined and 54 simulation design points per iteration were evaluated for each of the seven load cases. The value of 54 design points per iteration was slightly less than the recommend value of 1.5 times of number of variables, but this made efficient use of the computing resource available.

Space filling was set as the point selection method for the first load case while the other load cases were set to duplicate. Hybrid simulated annealing was adopted as the optimization algorithm.

For the NV analysis, MD NASTRAN R3 was used. The capability to have a direct interface to NASTRAN from within LS-OPT was a recent addition to the software that we used for this optimization to analyze the NASTRAN output. LS-DYNA 971 MPP SP R3.2.1 was used for all the crash analyses. To analyze the results for the crash load cases, internally-developed standard scripts using METAPOST from Beta CAE were executed. The output from these scripts could then be directly read into LS-OPT.

4.3 *Application and Execution*

The optimization was accomplished by using LS-OPT version 4.1 revision 63726 on a Linux workstation. The criterion for convergence of the optimization was set such that the design parameters and the objective had to change less than 0.01 between iterations. After the optimization had run for 10 iterations, this criterion was not satisfied so the optimization was allowed to continue for two further iterations. After the 12th iteration was completed, the optimization job was stopped, since it was found that only one constraint was slightly violated, which did not have significance in physical crash performance.

At this point, the optimization was switched to discrete variables to determine the actual optimum gauges for each part. This was accomplished by first creating a compressed LS-OPT database using the tools portion of the task menu. Once this database was created and compressed, it was moved to a new directory and uncompressed. Now each variable was switched from a continuous to a discrete variable and the range of discrete values was carefully selected from the closest available thicknesses for the material used with that part. Re-optimization was started on the 12th iteration using "Repair->Optimize current iter" in the LS-OPT task pull-down menu, and then it was switched back to Metamodel-based optimization and run. Through this process, LS-OPT was able to use its already developed metamodels to predict the optimum design point with discrete variables – no further analysis was performed to find the optimum with discrete variables. Finally, LS-OPT performed verification runs for all the analysis cases as the only design point of the 13th iteration.

4.4 *Results*

The final optimal design met all of the performance requirements. The optimal design had a 6 % reduction in mass from the baseline design. The thickness changes of the optimized components were not uniform. The thicknesses of 21 components decreased, while thicknesses of 10 components increased. There were no thickness changes in 4 components. The thickness changes are shown in figure 9.

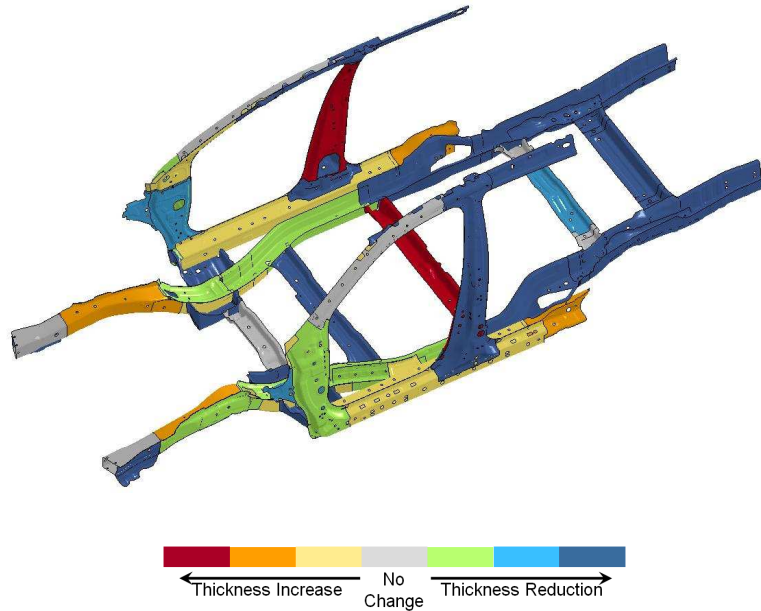


Fig. 9. Changes to thicknesses after the discrete portion of the SRSM MDO.

5 Conclusion

The testing of a variety of optimization methods on single-objective MDO, multi-objective MDO, and shape optimization problems found that RSM based methods performed consistently well. On the single objective problem, the sequential metamodeling method with domain reduction of LS-OPT showed better performance than any other method evaluated.

The development of the LS-OPT Virtual Machine and CScript language has provided the means to extend LS-OPT 4.1 into our enterprise environment. This implementation fits directly into our existing HPC infrastructure and reduces the complexity and workload to the user. Speed and efficiencies are realized by using multiple resources in parallel optimized for our practices. We successfully demonstrated that hundreds of optimization jobs can be managed by single or multiple users and only limited by the physical resources necessary for the computations.

When the SRSM method was applied to a much larger MDO problem it again showed good performance. An optimum design was found with mass savings that still met all of the constraints. This method within LS-OPT also showed good flexibility in being able to switch from continuous to discrete variables and recomputed an optimal solution using the available metamodels.

By including a MDO step within the BIW development process far more designs can be evaluated in a shorter period of time than could ever be accomplished through manually iteration. It does require increased computational resources to complete the MDO, but the potential mass savings justify the effort.

6 Acknowledgements

The authors would like to thank the LS-OPT developers from LSTC Nielen Stander and Trent Eggleston for their support during this activity

The authors would also like to thank the Ohio Super Computing Center for their assistance in providing the necessary computational resources required during portions of this activity.

7 References

- [1] Stander N, Roux W, Goel T, Eggleston T, Craig K. *LS-OPT User's Manual, Version 4.1*. Livermore Software Technology Corporation, Livermore, California, USA, December 2010.