STRUCTURAL OPTIMIZATION OF PRODUCT FAMILIES EXPOSED TO CRASH LOADING

AUTHOR:
Michael Öman
Department of solid mechanics Linköping University, Sweden and Scania CV AB
michael.oman@scania.com

Abstract

This paper discusses the problem of structural optimization in product family design and different methodologies are evaluated for weight optimization of components that are common for members in the family. First, the benefits of product family design are presented, followed by a discussion of the complexity of the design problem when the product family is exposed to crash loading and multiple load cases. It is concluded that in large problems the number of combinations of product and load case can get very large and due to the large number of simulations needed, it becomes practically impossible to solve the optimization problem with traditional methods. Therefore, a new methodology is presented that reduces the number of simulations by identifying the most critical combination of product and load case for every boundary condition and iteration. Different optimization methodologies are applied to a family of crash boxes and, finally, results and efficiencies are compared.

Introduction

The increasing competition in the automotive market is putting pressure on the vehicle manufacturers to offer the customers from different market segments a variety of products that satisfy the different requirements. To manage this at a reasonable cost, Meyer and Utterback [1] are arguing for a product family approach. The definition of a product family varies in the literature but the definition used in this article is as follows: A product family is a family of products where every product shares at least one component with at least one other product in the family. The product family approach is presently used by many companies to reduce manufacturing and development costs.

The drawback of the product family approach is that no individual product will be optimal. Every product will be built up of components that have to be designed for the requirements of other products in the family. This will be shown in the example later in this paper.

For a successful product family the number of shared components should be large. On the other hand, if too many components are shared by the products the range of customer requirements will not be covered by the product family. Many studies have been made trying to find out which components that are to be shared within the product

family. For example, D'souza and Simpson [2] and Nayak, Chen and Simpson [3] use different methods to optimize the product family and composition. The objective of the work behind this paper has been to optimize the weight of a product family, exposed to crash loading, where the composition of the components is already decided. That is, weight optimization of an already defined product family.

Crash loading and optimization

In the automotive industry today, crash safety is one of the most important requirements. The major part of the components in the car body is designed based on crash requirements. This makes structural optimization of the car body very expensive since crash simulations are computationally time consuming and the responses are noisy. The most common optimization method for crash loadings is therefore the Response Surface Methods, RSM. An approximate response surface is created from a limited number of evaluations and the optimum is then searched on the approximate surface. For every iteration, a new approximate surface is created and it is moved or changed in size depending on the results of the previous iteration. Some extensive work have been carried out by Redhe [4, 5] and Forsberg [6] evaluating different algorithms to span the approximate response surface. To span the simplest and cheapest linear surface, one simulation is needed for every design variable plus one base evaluation, i.e. n+1 simulations if n is the number of design variables. For every extra simulation performed, a surface that better represent the real response can be spanned. To choose which points that shall be evaluated, different design of experiment methods d.o.e. can be used. This paper uses Koshal d.o.e. and D-optimal d.o.e. to span linear response surfaces. The required number of simulations using the Koshal d.o.e. and linear surfaces is n+1, while the number of extra simulations using D-optimal d.o.e. and linear response surfaces recommended by Redhe [4] is 1.5 times the number used by the Koshal d.o.e. plus one, or ((n+1)*1.5)+1.

Optimization of product families

As mentioned above, no individual product in the family is optimal. What is interesting from a cost efficiency point of view, is the optimality of the product family as a whole. The difference in optimizing the entire product family instead of only one product is that the requirements of all products have to be taken into account for every component. This fact makes optimization of product families complicated and expensive to perform. For every iteration and design variable, function evaluations have to be done for every product in which the variable is present. This is of minor importance if the function evaluations are cheap but a big issue if the evaluations are expensive. An example of an optimization of cheap function evaluations can be found in the work by Torstenfeldt and Klarbring [7]. They perform a weight optimization of a product family subjected to linear loads. In the case of a car body where the dominant load cases are crash

requirements, the number of evaluations is of great importance. As mentioned, optimization of one product exposed to one crash requirement is expensive. In the case of a product family, the number of evaluations needed for every iteration grows for every product and requirement added. Even for a family with only a few products, requirements and design variables, the problem becomes unrealistically large for the existing optimization methods available today. This is why a new algorithm is presented in the following chapter.

Simplified optimization algorithm

The basic idea of this simplified optimization algorithm is to perform just the most relevant function evaluations. This is done by identifying the most critical product for every requirement and then only take the identified combinations of products and load cases into account in the optimization. The idea is that every design variable shall only be optimized in the combination of product and load case where the variable is most active. The design variables are therefore divided between these identified critical combinations. Design variables not present in the identified critical combinations will be optimized elsewhere, in the combination where they are present and most active. The algorithm is described by the following steps.

Simplified algorithm

A function evaluation is performed for every combination product and load case with the initial design variable values.

Then, for every constraint or requirement the most critical combination of product and load case is identified. That is, the combination giving a response value closest to or violating the constraint value the most.

Next, the design variables present in the identified critical combinations of product and load case are divided between these combinations. The variables will only be optimized in the combination where they are most active. This is for example done by evaluating the internal energy. For example, variable x is most active in combination y, and is therefore given to combination y.

One iteration is now performed for the critical combination that has been given the most design variables, using LS-Opt.

The values of the optimized design variables are updated before the next combination is optimized.

Next, the critical combinations are optimized one by one in the order of number of given design variables, using the updated values of the variables form the earlier optimized combinations.

The design variables not yet optimized are now divided between the rest of the combinations of products and load case. Every variable is given the combination where it is most active, i.e. using the internal energy.
The design variables are then optimized one iteration, starting with the combination with the most given variables and updating the values of the design variables after every optimization.
Now every design variable has been optimized one iteration but before starting the next iteration the size of the response surface, called the region of interest, for every design variable is evaluated and updated for the next iteration.
We can now start the second iteration by going back to step no. 1, performing only the evaluations of the combinations not yet evaluated with the actual design variable values in the present iteration.

In this way, the number of function evaluations necessary per iteration is reduced. How much depends on the formulation of the product family. To evaluate this simplified algorithm, a program was written in Python which calls the optimizing program LS-Opt.

During the work behind this paper, a problem with the simplified algorithm was identified. When the critical combination of product and load case is changed from one iteration to another, the algorithm goes back to its earlier point of the last iteration and tends to get trapped jumping back and forth. The reason is that no information is transferred from the earlier iteration. To get around this problem the common design variables in the old and new critical combination were optimized in both combinations. The drawback is that this results in more simulations per iteration.

Example

To test the simplified optimization algorithm a fictive family of crash boxes was constructed. The family consists of four boxes of different size and shape, two quadratic boxes, one small and one large and two rectangular boxes, one with and one without midsection, see Figure 1. The boxes are built out of seven different components with their thicknesses as design variables and placed standing fixed to the ground. The boxes are all exposed to two load cases, one impact loading in the vertical direction and one quasi-static loading in the horizontal direction. The boxes are allowed to deform a maximum of 20 mm due to the impact load and the horizontal load has to result in an internal energy in the box of minimum 450 kJ. The objective is to minimize the total weight of the family of boxes.

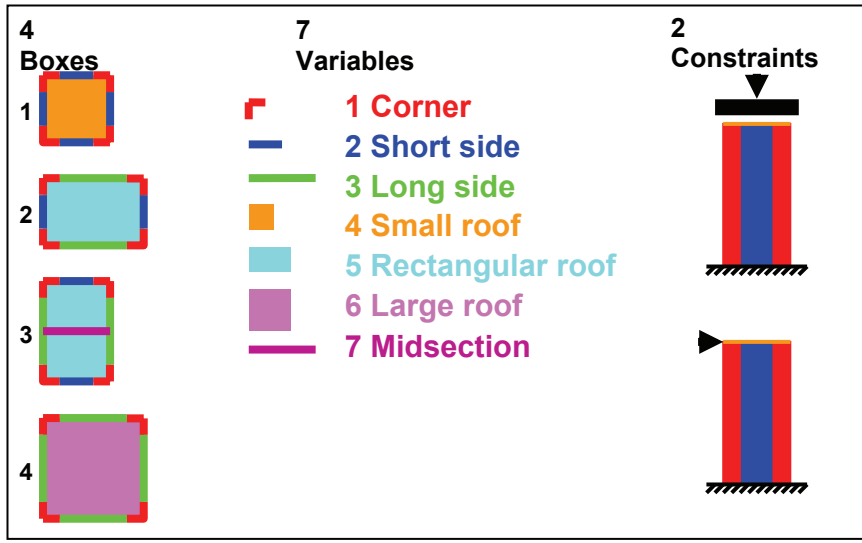

Figure 1: Schematic overview of the product family of crash boxes

Using the traditional optimization method and the cheapest point selection scheme, linear response surfaces with Koshal d.o.e., a total of 42 simulations is necessary per iteration. The more expensive D-optimal d.o.e. with over-sampling will need 64 simulations. The number of simulations per iteration using the simplified algorithm will vary from iteration to iteration, but will be about 20 simulations using the Koshal d.o.e. and about 30 simulations using the D-optimal d.o.e.

To be able to make fast evaluations, small models were built. The element size is 5.0 mm and the four different boxes contain only 516, 700, 820 and 900 elements. With a simulation time of 10 ms one simulation takes 13-25 seconds depending on the load case and crash box. This makes an evaluation possible because an optimization using about 500 simulations will then terminate after about 10.000 seconds, or 2 hours and 45 minutes. A more realistic model and load case would take much more then 20 seconds. As an example a solution time of one hour results in a total optimization time of 21 days.

Results

First, the extra total weight of the four boxes for using common components was studied with the following results. If the entire product family is optimized instead of every individual box, the total weight of the boxes increases with 11.4% using LS-Opt and Koshal d.o.e. and 11.9% using D-optimal d.o.e. Using D-optimal d.o.e. both the total weight of the boxes in the single optimizations and the weight of the product family decreases compared to the Koshal d.o.e. This is due to the better optimum found using the D-optimal d.o.e. The initial thicknesses of all design variables were 1.0 mm. The results are summarized in Table 1.

D.o.e. method	Koshal		D-optimal	
Optimization	Single box	Product family	Single box	Product family
Weight box 1	0.1610	0.1894	0.1613	0.1827
Weight box 2	0.2060	0.2201	0.2053	0.2166
Weight box 3	0.1833	0.2318	0.1775	0.2284
Weight box 4	0.2436	0.2432	0.2425	0.2524
Total weight	0.7939	0.8845	0.7866	0.8801
Weight increase	11.4 %		11.9 %	

Table 1: The results from the weight optimizations of product by product and the entire product family

Next the efficiency of the simplified algorithm was evaluated. The family of crash boxes was optimized both using Koshal d.o.e. and D-optimal d.o.e.. Two different starting points were also tested to see if the same conclusions could be drawn. First, the thickness of all design variables was set to 1.0 mm and secondly to 2.0 mm. All optimizations were set to terminate when the change in design variables and the change of the objective function were less than 0.01.

It was found that the simplified optimization algorithm does converge more rapidly than the traditional algorithm. As presented in Table 2 the simplified algorithm using Koshal d.o.e. is the fastest to converge. It uses 272 simulations starting with a thickness of 1.0 mm and 290 simulation starting at 2.0 mm. The number of simulations used by the traditional method using Koshal d.o.e. is 518 starting at 1.0 mm and 474 simulations starting at 2.0 mm. The reduction in simulations is 48% and 39% starting at 1.0 and 2.0 mm. If the D-optimal d.o.e. is chosen the reduction is in the same order, 56% and 40%, compared to the traditional method using D-optimal d.o.e. Two different optima with similar results were found in the different optimizations. The better optima giving a total weight of about 0.878-0.880 kg were found by the traditional method using D-optimal d.o.e. starting at 1.0 mm and by the simplified algorithm using D-optimal d.o.e. starting at 2.0 mm. All other optimizations found the second optima giving a total

weight of about 0.885-0.892 kg. The differences between the two optima can be seen looking at the values of design variable 1, 2 and 5 in Table 2.

What maybe is more important in an industrial case where the number of possible simulations often is limited due to long computational times, is that a reasonable good result is obtained even if the optimizer is stopped early. It is therefore important that the optimizer gets close to the optimum rapidly. To study this, the optimization history is drawn in diagrams 1 and 3, for all optimizations in Table 2. Amplifications of the interesting areas are drawn in diagrams 2 and 4. The simplified algorithm using Koshal d.o.e. proved to be the method that gets close to the optimum most rapidly. In the two cases, starting at 1.0 mm and 2.0 mm, diagrams 1 and 3, a reasonably good result is obtained already after about 100 simulations. In general using Koshal d.o.e. seems to be the best choice for optimization of product families. The methods using D-optimal d.o.e. generally does find better optimums, but the methods using Koshal d.o.e. need fewer simulations to get close to the optimum and they converge faster. This is not the case for an optimization of one product. According to Redhe [4], using D-optimal d.o.e. gives the best results for a single product and single load case optimization.

Starting value	1.0 mm				2.0 mm			
Optimization method	LS-Opt		Simplified algorithm		LS-Opt		Simplified algorithm	
	Koshal	D-opt.	Koshal	D-opt.	Koshal	D-opt.	Koshal	D-opt.
No. of iterations	13	11	13	11	12	11	13	14
No. of simulations	518	712	272	314	474	712	290	428
1 Corner [mm]	2,451	2,812	2,401	2,451	2,351	2,596	2,438	2,815
2 Short side [mm]	1,733	1,441	1,776	1,773	1,806	1,605	1,747	1,436
3 Long side [mm]	1,243	1,223	1,252	1,256	1,255	1,246	1,24	1,216
4 Small roof [mm]	0,510	0,549	0,505	0,500	0,508	0,536	0,507	0,533
5 Rectangular roof [mm	0,902	0,534	0,939	0,871	0,967	0,776	0,926	0,520
6 Large roof [mm]	0,517	0,534	0,597	0,500	0,566	0,511	0,521	0,543
7 Midsection [mm]	0,500	0,500	0,500	0,500	0,500	0,500	0,500	0,500
Box 1 intrusion [mm]	15,44	16,27	15,31	15,14	13,69	14,42	15,38	16,21
Box 1 energy [kJ]	449,88	450,99	449,81	454,60	450,20	449,31	450,10	445,43
Box 2 intrusion [mm]	20,04	19,97	19,97	19,69	20,17	20,00	20,00	20,01
Box 2 energy [kJ]	607,00	450,14	630,40	599,77	645,45	545,93	617,59	441,68
Box 3 intrusion [mm]	14,04	14,27	13,93	14,20	13,13	13,06	13,94	14,11
Box 3 energy [kJ]	684,77	517,93	687,97	690,24	687,73	630,37	685,74	512,69
Box 4 intrusion [mm]	19,77	19,68	19,31	19,25	20,46	20,10	19,21	19,39

Box 4 energy [kJ]	450,08	450,23	478,84	450,94	469,61	450,34	450,06	450,04
Weight box 1 [kg]	0,1894	0,1827	0,1905	0,1918	0,1908	0,1861	0,1898	0,1823
Weight box 2 [kg]	0,2201	0,2166	0,2208	0,2218	0,2207	0,2193	0,2202	0,2160
Weight box 3 [kg]	0,2318	0,2284	0,2326	0,2335	0,2325	0,2311	0,2320	0,2277
Weight box 4 [kg]	0,2432	0,2524	0,2444	0,2445	0,2426	0,2480	0,2425	0,2518
Total weight [kg]	0,8845	0,8801	0,8883	0,8916	0,8866	0,8845	0,8845	0,8778

Table 2: Summary of the results of the product family optimizations

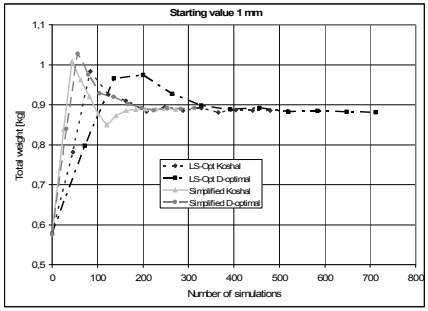

Diagram 1: Optimization history of the different methods from Table 2 with starting value 1.0 mm

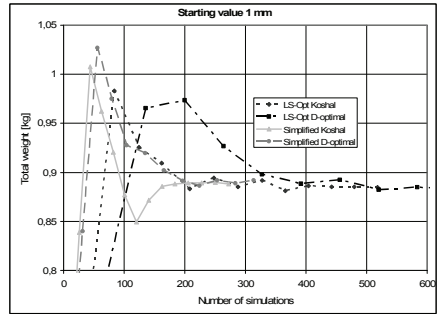

Diagram 2: Enlargement of the interesting area of Diagram 1

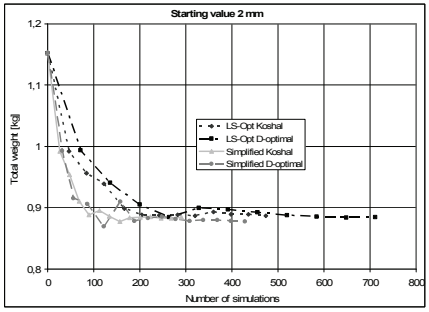

Diagram 3: Optimization history of the different methods from Table 2 with starting value 2.0 mm

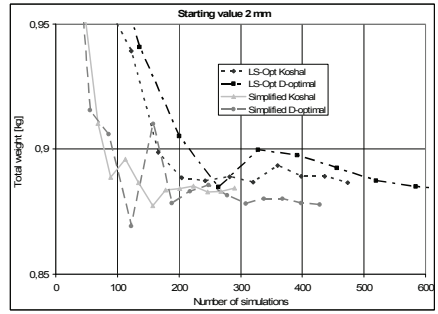

Diagram 4: Enlargement of the interesting area of Diagram 3

Conclusions

A simplified family optimization problem has been presented. However, the reduction of simulations using the simplified algorithm of about 50% was a bit disappointing. A larger reduction was expected and maybe necessary to make the optimization of product families exposed to crash loading more realistic, but the algorithm can be made more efficient. Allowing transferring of information between the iterations may be a possible way of improving the efficiency. It might also be more efficient not to search for a new critical combination of product and load case in every iteration. These and other modifications will possibly reduce the number of simulations per iteration. One other thing affecting the efficiency of the algorithm is the composition of the product family. The more components or design variables that are shared within the family, the more efficient will the simplified algorithm be.

The simplified algorithm has only been tested on a simple product family, and thus no general conclusions can be drawn. The results still show that the simplified algorithm has a potential and encourages further refinements and investigations.

References

[1] **Meyer, M. H. and Utterback, J. M.**, The product family and the dynamics of core capability, *Sloan Management Review*, 34, 29-47, 1993.

[2] **D'souza, B. and Simpson, T. W.**, A generic algorithm based method for product family design optimization, *Engineering Optimization*, Vol 35(1), pp. 1-18, 2003.

[3] **Nayak, R. U., Chen, W. and Simpson, T. W.**, A variation-based method for product family design, *Engineering Optimization,* Vol 34(1), pp. 65-81, 2002.

[4] **Redhe, M., Forsberg, J., Jansson, T., Marklund, P-O. and Nilsson, L.**, Using the response surface methodology and the D-Optimality criterion in crash-worthiness related problems – an analysis of the surface approximation error versus the number of function evaluations, *Structural and Multidisciplinary optimization*, Vol 24, issue 3, pp. 185-194, 2002.

[5] **Redhe, M., Jansson, J. and Nilsson, L.**, Using surrogate models and response surfaces in structural optimization, *Structural and Multidisciplinary optimization*, Vol 25, issue 2, pp. 129-140, 2003.

[6] **Forsberg, J.**, Simulation based crashworthiness design – Accuracy aspects of structural optimization using response surfaces, LIU-TEC-LIC-2002:27, Linköping University, Linlöping, 2002.

[7] **Torstenfelt, B., Klarbring, A.**, Structural optimization of modular product families with application to car space frame structures, *Structural and Multidisciplinary optimization,* Vol 32, issue 2, pp. 133-140, 2006.