Optimal Sizing using Automatic Differentiation

Wolfram Mühlhuber *
Spezialforschungsbereich SFB F013
"Numerical and Symbolic Scientific Computing"
Freistädterstr. 313, 1. Stock
A - 4040 Linz, Austria
email: wmuehlhu@sfb013.uni-linz.ac.at

Abstract

This paper is devoted to optimal sizing. As a model example we consider the minimization of the mass of the frame of an injection moulding machine. The deformation of the frame is described by a generalized plane stress state with an elasticity modulus scaled by the thickness. This constrained nonlinear optimization problem is solved by sequential quadratic programming (SQP) which requires gradients of the objective and the constraints with respect to the design parameters.

As long as the number of design parameters is small, finite differences may be used. In order to handle also several hundreds of varying thickness parameters, we use the reverse mode of automatic differentiation for differentiating the function evaluation. This approach works fine but requires huge memory and disk capabilities. Furthermore, the use of iterative solvers for the governing state equations is limited. Therefore, we combine it with the adjoint method to get a fast and flexible gradient evaluation procedure.

The presented numerical results show the potential of this approach and imply that this method can also be used for finding an initial guess for a shape optimization.

Keywords: optimal sizing, shape optimization, finite elements, automatic differentiation, adjoint method

AMS Subject Classification: 73K40, 90C90, 49Q10

1 Introduction

The design of a machine or of a mechanical structure has to fulfill various constraints in many industrial applications. In most cases, an optimal design subject to several restrictions is desired. These restrictions are mainly induced by technological limits, requirements of the individual customer or indirectly by competition on the world market. Unfortunately, due to lack of time, engineers

*This work was supported by the Austrian Science Fund - 'Fonds zur Förderung der wissenschaftlichen Forschung (FWF)' - SFB F013 'Numerical and Symbolic Scientific Computing'

designing a machine component have to stop their design process after a few iterations – in most cases only two or three. Then they take the best design obtained so far because no more time is left for drafts that would possibly meet the requirements to a larger extent.

Therefore, tools accelerating such a design process have to fulfill mainly two goals:

- They have to be flexible enough to handle the various requirements. Nevertheless, they also have to be robust to produce reliable results. Especially, it is desirable to spend only little work on modifying the code when the requirements change.
- On the other hand, these tools have to be fast. The faster the tool, the more design drafts can be optimized.

An extensive review of the various methods for structural optimization using finite elements is given in the monograph of Haslinger, Neittaanmäki [8]. A monograph specializing more on topology optimization was presented by Bendsøe [1]. Both contain many references on the various aspects of shape optimization in general. Mahmoud [10] and Mahmoud, Engl, Holzleitner [11] focus on an optimal sizing approach similar to the one in this paper, but they use approximate representations of the objective and of the constraints to reduce the overall costs of the method. Stangl [17] presents a generalization of that approach to a class of nonlinearly elastic materials. Some stability results with respect to an inexact solution of the state equation are given in Stangl [16].

Petersson [13] presents connections between stable discretizations for a material distribution problem and stable ones for Stokes' problem. An efficient solution strategy for this problem was presented by Maar, Schulz [9]. For approaches using a topology optimization for getting an initial guess of the topology used in a shape optimization afterwards see e.g. Maute, Ramm [12] or Ramm, Bletzinger, Reitinger, Maute [15].

This paper deals with minimizing the mass of the frame of an injection moulding machine as an example for a typical optimal sizing problem. After modeling the problem, we show how to transform it into an equivalent one which is easier manageable by the optimizer. Furthermore, a short sketch on the used optimization method is presented. The main problem in applying standard optimization methods is to calculate gradients of the given objective and of the restrictions. Since implementing analytic derivatives is an improper approach, various alternatives are presented and compared to each other. Besides finite differences, a very flexible approach using automatic differentiation is presented. But in order to get also an efficient and fast method, automatic differentiation has to be coupled with a well-known approach from shape optimization – the so-called adjoint method. Numerical results show the strength of this approach.

The paper is organized as follows: The problem is modeled and the arising optimization problem is stated in Section 2. Section 3 deals with the handling of the problem from the optimization's point of view. Section 4 presents some

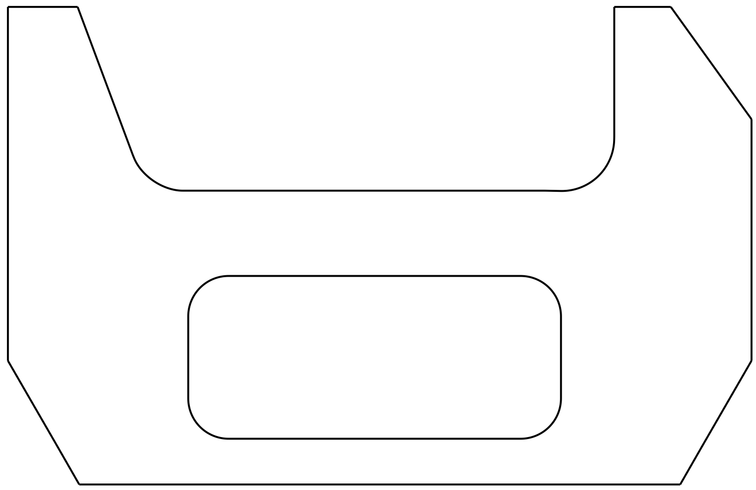

Figure 1: Cross section of the original shape

methods for calculating gradients of the given objective or the restrictions. In Section 5, numerical results are presented.

2 The modeling of the design of the frame of an injection moulding machine

The frame of an injection moulding machine is briefly sketched by its 2D-cut Ω given in Figure 1. For a frame of homogeneous thickness, typical dimensions are:

- thickness of one plate: 180 mm
- mass of one plate: 3.8 tons
- clumping force (surface force): 300 tons $\approxeq$ 16 N/mm^2
- length: 2.8 m
- height: 1.7 m
- 2 supporting areas

The primary goal of the design phase is to minimize the mass of the frame of the injection moulding machine. Several other requirements have to be fulfilled in addition, e.g.

- maximal v. Mises stress: $\sigma^{\mathrm{vM}} \leq \sigma^{\mathrm{vM}}_{\max}$,
- maximal tensile stress: $\sigma^{\mathrm{ten}} \leq \sigma^{\mathrm{ten}}_{\max}$,
- shrinking angle of the clumping unit (vertical edges on top, called *wings*): $\alpha \leq \alpha_{\max}$,

- handling of the machine or the feeding mechanism,
- easy and cost efficient manufacturing.

Some of these constraints can be integrated into the optimization procedure directly (e.g. restrictions on the stresses), whereas others like the easy manufacturing have to be considered in a post-processing step.

In order to evaluate the stresses σ, the displacement field u of the frame under some load F has to be known.

For a fixed thickness $\rho(x)$, the displacement field $u(x), x \in \Omega$, fulfills

$$a(\rho; u, v) = F(v) \qquad \text{for all } v \in V_0 \tag{1}$$

with

$$a(\rho; u, v) = \int_\Omega \rho \, \frac{\partial u_i}{\partial x_j} \, E_{ijkl} \, \frac{\partial v_k}{\partial x_l} \, dx, \quad F(v) = \int_\Omega \langle f, v \rangle \, dx + \int_{\Gamma_N} g \, v \, ds$$

where E_{ijkl} denotes the elasticity tensor, f the volume force density and g the surface force density on a part Γ_N of the boundary. u and v are assumed to be in $V_0 = \left\{ v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_D, \text{meas } \Gamma_D > 0 \right\}$ (set of admissible displacements) where $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$.

E_{ijkl} is given by

$$E_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}), \tag{2}$$

where δ_{ij} denotes the Kronecker - Delta and λ, μ denote Lame's constants. These can be calculated from Young's modulus E and Poisson's ratio ν by

$$\lambda = \frac{E \, \nu}{(1+\nu)(1-\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}. \tag{3}$$

In (1) we have assumed that

- a generalized plane stress problem is considered, i.e. we consider a body Ω as a plate that is thin in x_3-direction (compared to the other coordinate directions) and that can carry stresses only parallel to the x_1-x_2-plane, and
- the applied surface tractions and the body forces are independent of x_3, and therefore, there is no displacement in x_3-direction and the other two displacement components are also independent of x_3.

The design problem can be stated as follows:

$$\begin{aligned} & & \int_\Omega \rho \, dx \rightarrow \min_{u,\rho} & & \\ & \text{subject to} & a(\rho; u, v) = F(v) & & \text{for all } v \in V_0 \\ & & 0 < \underline{\rho} \le \rho \le \overline{\rho}, & & \text{a.e. in } \Omega \\ & & \sigma^{\text{vM}}(u) \le \sigma^{\text{vM}}_{\text{max}} & & \text{a.e. in } \Omega, \\ & & \sigma^{\text{ten}}(u) \le \sigma^{\text{ten}}_{\text{max}} & & \text{a.e. in } \Omega \\ & & \alpha(u) \le \alpha_{\text{max}} & & \end{aligned} \tag{4}$$

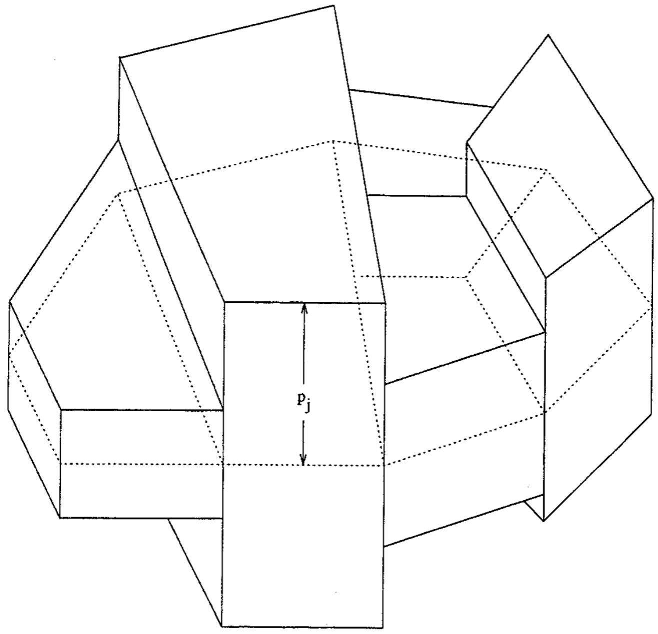

Figure 2: Sizing: the design variables are the thicknesses ρ_j of the subregions

$\sigma^{\mathrm{vM}}(u)$ denotes the v. Mises stress, $\sigma^{\mathrm{ten}}(u)$ the tensile stress in the frame. The change in the shrinking angle of the clumping unit (vertical edges on top, called *wings*) is denoted by $\alpha(u)$.

For discretizing the problem, we use triangular finite elements with piece-wise constant shape functions for approximating ρ and piece-wise quadratic ones for approximating u. In many situations it is additionally assumed that ρ is constant in certain non overlapping subregions Ω_i (c.f. Figure 2).

We denote the discrete approximation of ρ and u again by ρ and u. Clearly, the situation in Figure 2 would yield to singularities in the stress distribution for the continuous problem, but the piecewise constant thickness distribution has to be understood only as an approximation for a continuous thickness distribution over the whole cross-section (see e.g. HASLINGER, NEITTAANMÄKI [8], MAHMOUD [10] or STANGL [17]).

Summarizing all those considerations, the discretized optimization problem can be formulated as follows:

$$
\begin{aligned}
\operatorname{mass}(\rho) = \int_{\Omega_h} \rho(x)dx \rightarrow \min_{\rho}& \\
K(\rho)\, u = F(\rho)& \\
\sigma^{\mathrm{vM}}(u) \leq \sigma^{\mathrm{vM}}_{\max}& \\
\sigma^{\mathrm{ten}}(u) \leq \sigma^{\mathrm{ten}}_{\max}& \\
\alpha(u) \leq \alpha_{\max}& \\
\underline{\rho} \leq \rho \leq \overline{\rho},&
\end{aligned}
\tag{5}
$$

with the symmetric positive definite, large, sparse stiffness matrix K. For the discretized problem the constraints on σ^{vM}, σ^{ten} and ρ have to be understood component-wise.

In our application, the upper limits on the angle and the stresses are either treated as constraints or as soft limits, which can be violated to some extent, if the mass would be severely smaller then. Furthermore, the pointwise constraints on σ^{vM} and σ^{ten} are replaced by using a higher order ℓ^p norm. Treating the upper limits as soft constraints leads to the following reformulation:

$$
\begin{aligned}
&\text{mass}(\rho) + \omega_1\big(\max\left(\|\sigma^{\text{vM}}\|_p - \sigma^{\text{vM}}_{\text{max}}, 0\right)\big)^2\\
&\qquad + \omega_2\big(\max\left(\|\sigma^{\text{ten}}\|_p - \sigma^{\text{ten}}_{\text{max}}, 0\right)\big)^2\\
&\qquad + \omega_3\big(\max\left(\alpha - \alpha_{\text{max}}, 0\right)\big)^2 \qquad \to \min_{u,\rho}\\
&\qquad K(\rho)\, u = F(\rho)\\
&\qquad \underline{\rho} \le \rho \le \overline{\rho}
\end{aligned}
\tag{6}
$$

where ω_i denote user-chosen factors of influence. Note that this modification leads to an objective in C^1. It looks similar to a penalty formulation of the constraints, but the weights ω_i are kept fixed during the iteration and do not tend to infinity like in penalty methods. Therefore, the problem of ill-conditioning of the Hessian is avoided.

3 A review from the optimizer's point of view

From the optimization's point of view the problem (6) reads as

$$
\begin{aligned}
& && J(u,\rho) \to \min_{u,\rho}\\
&\text{subject to} && K(\rho)\, u = F(\rho)\\
& && \underline{\rho} \le \rho \le \overline{\rho},
\end{aligned}
\tag{7}
$$

where ρ denotes the vector of design parameters and u the solution of the governing finite element (FE) state equation. The splitting of the parameter vector into design parameters ρ and the solution of the FE equation u is typical for problems in shape optimization. From the optimization's point of view the discretized state equation can be interpreted as equality constraints. In our case it is linear with respect to u and $K(\rho)$ is symmetric and positive definite for all admissible parameters ρ. Therefore, eliminating u formally, we arrive at

$$
\begin{aligned}
& && \tilde{J}(\rho) = J(K^{-1}(\rho)\, F(\rho),\, \rho) \to \min_{\rho}\\
&\text{subject to} && \underline{\rho} \le \rho \le \overline{\rho}.
\end{aligned}
\tag{8}
$$

Since we want to use a standard SQP method for optimizing, the formulation (8) is advantageous compared to (7) as it has much fewer parameters. This relies on the fact, that our implementation of the SQP method is based on linear algebra with dense matrices. A short sketch onto a model SQP algorithm for solving

$$\begin{aligned} & \tilde{J}(\rho) \to \min_{\rho} \\ \text{subject to} \qquad & c(\rho) \leq 0 \end{aligned} \tag{9}$$

is given in Algorithm 1. Details can be found e.g. in FLETCHER [3] or GILL, MURRAY, WRIGHT [4].

Algorithm 1 SQP model algorithm

$B_0 = I$
$g_0 = \operatorname{grad} \tilde{J}(\rho_0)$
/* *linearize constraints,* $\doteq$ *denotes a first order approximation* */
$c(\rho_0 + \delta) \doteq A_0 \delta + b_0$
$k = 1$
while not converged **do**
  /* *calculate search direction* s_k */
  Solve
    $\frac{1}{2} s^T B_k\, s + g_k^T s \to \min$
  under the constraints $A_k s \leq -b_k - A_k \rho_k$
  /* *line search procedure* */
  Calculate $\alpha_k \in (0, 1]$ as large as possible such that
    $\Phi_k(\rho_k) + \mu_1 \alpha_k \Phi_k'(\rho_k) \leq \Phi_k(\rho_k + \alpha_k s_k) \leq \Phi_k(\rho_k) + \mu_2 \alpha_k \Phi_k'(\rho_k)$
    with $0 < \mu_1 < \frac{1}{2} < \mu_2 < 1$ with a suitable merit function Φ_k
  /* *Update several quantities* */
  $\rho_{k+1} = \rho_k + \alpha_k s_k$
  $g_{k+1} = \operatorname{grad} \tilde{J}(\rho_{k+1})$
  $c(\rho_{k+1} + \delta) \doteq A_{k+1} \delta + b_{k+1}$
  Update Hessian matrix $\to B_{k+1}$
  $k = k + 1$
end while

The optimizer used in our code is based on a Quasi-Newton approximation of the Hessian using a modified BFGS update formula following POWELL [14] in order to avoid the need for Hessian information of the objective. The quadratic subproblem is solved by a range space based QP method combined with an active index set strategy. The line search procedure uses an exact penalty function

$$\Phi_k(\rho) = \tilde{J}(\rho) + \sum_j \overline{\sigma}_j^k \max(\rho_j - \overline{\rho}, 0) + \underline{\sigma}_j^k \max(\underline{\rho} - \rho_j, 0) \tag{10}$$

with suitable chosen penalty parameters $\overline{\sigma}_j^k$, $\underline{\sigma}_j^k$ as merit function (c.f. HAN [7]).

4 Calculating gradients

Using a Quasi-Newton strategy and update formulas within the SQP method as proposed in the section before, the remaining main problem is the calculation of gradients for the objective and the constraints. As in this paper, the problem was transformed into a problem with box constraints only, routines providing analytic gradients for these constraints can be implemented easily. But for the objective, the implementation of an analytic derivative is by far too complicated and time consuming. Furthermore, it would not be well suited for the use in a design process, as we would loose the flexibility of the code completely. That is the reason why we have to think of alternative methods for calculating the gradients.

Several methods are presented and compared to each other in this section. On the one hand we have black box methods like finite differences or automatic differentiation (c.f. Griewank [5]), on the other hand, methods exploiting the special structure of the state equation are available, e.g. the direct method or the adjoint method (c.f. Haslinger, Neittaanmäki [8]).

As none of these methods is really well suited for our problem, a hybrid method combining automatic differentiation and the adjoint method is developed.

4.1 Finite differences

If no analytic derivatives of a function can be implemented due to the high complexity then an approximation by finite differences is often the first idea. One approximation is the central difference quotient

$$\mathrm{D}\, f(x) = \frac{f(x+h) - f(x-h)}{2h}. \tag{11}$$

The choice of the increment h is rather critical for getting good results and depends on estimates of the third derivative of f.

In order to improve the accuracy of finite differences, extrapolation methods can be used. For an initial increment H the sequence

$$\mathrm{D}\, f(H), \mathrm{D}\, f(\frac{H}{2}), \mathrm{D}\, f(\frac{H}{4}), \ldots, \mathrm{D}\, f(\frac{H}{2^i}), \ldots \tag{12}$$

is calculated and extrapolated for $i \to \infty$. These methods return not only a value for the derivative, but also an estimate on the precision of that value which can be used for controlling the order of the extrapolation scheme(c.f. Stoer [18]).

The main properties are summarized as follows:

- Two function evaluations are needed per difference quotient and in most cases several difference quotients are needed in order to reach the desired precision of the derivative. Furthermore, the number of function evaluations is proportional to the number of design parameters. Due to the high effort finite differences are only well suited for problems with few design parameters.

- Finite differences can easily be used for very complex functions, as they do not rely on any special properties. On the other hand they neither can use any special properties which makes the use of finite differences rather inefficient in certain cases.

- From the user's point of view finite differences are very flexible. Changes in the desired objective imply only a re-implementation of the objective function. Time consuming changes of the gradient routine do not appear, which is especially important for the acceptance of such a method in an industrial design process.

- The possible use of iterative methods for solving the state equation is very important for our problem as the number of parameters in the state equation may be rather large. As finite differences do not rely on any special properties of the function the coupling with iterative solvers can be done without any problems.

4.2 Automatic differentiation

Compared to finite differences, AD follows a completely different approach. Finite differences try to approximate the derivative and therefore do not provide accurate results, whereas AD methods incur no truncation error at all and usually yield derivatives with working accuracy. Starting point is a computer program that calculates numerical values for a function. First, a symbolic evaluation graph mapping the design parameters to the function values is built. Like symbolic differentiation, AD operates by systematic application of the chain rule, familiar from elementary differential calculus. However, in the case of AD, the chain rule is applied not to symbolic expressions, but to actual numerical values. By using all the intermediate results generated by the function evaluation, the exponential growth of the evaluation complexity of symbolic differentiation can be avoided, as many common subexpressions can be used repeatedly when the gradient is evaluated at any particular point. Furthermore, optimizations made for the function evaluation also pay off for its derivative. Details on how the AD technique works, as well as many related issues on calculating higher order derivatives can be found in Griewank [5].

Two different kinds of tools are known in the AD community: The first group is based on source code transformation, e.g. ADIFOR written for FORTRAN codes, c.f. Bischof et al. [2]. The other group is based on evaluation graphs generated at runtime, e.g. ADOL-C written for C and C++ codes, c.f. Griewank, Juedes, Utke [6]. As our finite element code is completely written in C++ and uses heavily virtual inheritance, source code transformation tools can not be used. It has to be mentioned, that some of the properties of AD listed in the following rely on the use of runtime tools.

- ADOL-C needs a file containing the evaluation graph in a symbolic form for evaluating the function and its gradient. This file is generated at runtime. For structural optimization problems, huge memory and disk capabilities are required for that purpose. Due to the need of an evaluation graph, ADOL-C can only be applied to functions of moderate complexity. The limiting factor is not the inherent complexity of the function itself, but the size of the generated files and the time needed for reading and writing the data. To give an example, the files storing the evaluation graph for a problem with about 450 design parameters and about 7500 DOFs in the FE state equation need about 1 GB of disk space.

- The flexibility of AD with respect to changes in the objective is similar to finite differences. Changes in the objective need only a re-implementation of the objective function, but no changes in the gradient routine when runtime tools are used. Sometimes, special care has to be taken for a correct generation of the evaluation graph, especially in the context of conditional statements.

- AD using ADOL-C is a black box method. The use of the evaluation graph is a drawback of the method, especially when debugging is needed. This is compensated to some extent by the good runtime behaviour of the method. For the so-called reverse mode, the calculation time of the gradient is independent of the number of design parameters and takes the time of about 15 - 20 native C++ function evaluations as long as the evaluation graph can be stored in the main memory of the computer. Compared to the use of finite differences, this is a tremendous speedup, even for problems with only 10 - 20 design parameters.

- The coupling of AD with iterative solvers is a problem of current research (see e.g. Griewank [5] and references therein). As the use of iterative methods (e.g. multilevel methods) is important for solving fine FE discretizations of the state equation efficiently, the applicability is limited to problems, where direct solvers can be used.

4.3 Direct and Adjoint Method

Both methods are well-known in the shape optimization community (see e.g. Haslinger, Neittaanmäki [8]) and take into account the special structure of the state equation. They differentiate the state equation with respect to a design parameter ρ_i leading to

$$K\,\frac{\partial u}{\partial \rho_i} = \frac{\partial F}{\partial \rho_i} - \frac{\partial K}{\partial \rho_i}\,u. \tag{13}$$

For the direct method, (13) is solved numerically using the same methods as for the state problem itself. Then the gradient of the objective can be calculated by

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\rho_i} = \frac{\partial J}{\partial \rho_i} + \langle \frac{\partial J}{\partial u}, \frac{\partial u}{\partial \rho_i} \rangle. \tag{14}$$

On the contrary to the direct method, the adjoint method solves (13) formally and inserts the result in (14) which leads to

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\rho_i} = \frac{\partial J}{\partial \rho_i} + \langle K^{-T}\,\frac{\partial J}{\partial u}, \frac{\partial F}{\partial \rho_i} - \frac{\partial K}{\partial \rho_i}\,u \rangle \tag{15}$$

In the following the main properties are summarized:

- The solution of one state equation per design parameter is needed for the direct method, whereas the adjoint method needs the solution of one adjoint problem for the objective and in principle for each constraint. As K is symmetric in our case, the effort for solving the adjoint problem is the same as for solving the state equation itself. Depending on the number of design parameter and constraints, the better suited method can be chosen.

- As analytic partial derivatives of J with respect to ρ and u are needed ($\frac{\partial J}{\partial \rho}$, $\frac{\partial J}{\partial u}$, $\frac{\partial K}{\partial \rho}$, $\frac{\partial F}{\partial \rho}$), both methods can only be applied to simple objectives, where this can be done easily. Furthermore, the flexibility of the method suffers from the need of hand-coded gradient routines.

- Compared to finite differences or the use of AD for the whole function, this approach is much faster. Finite differences need much more solutions of the design problem, compared to AD the huge evaluation graph which originates mainly from the solution of the state equation is avoided.

- Any solver can be used for solving the state problem, especially the use of iterative solvers like conjugate gradient methods with multilevel preconditioning is no problem.

4.4 Hybrid Method

Comparing the last two sections, it can be seen that the strengths of these methods lie in completely different areas. AD provides very high flexibility with respect to the used objective, but has drawbacks with respect to the needed computer requirements, the use of iterative solvers for the state equation and with respect to longer runtime. On the other hand the direct and the adjoint method can easily be combined with iterative solvers and provide a fast way for calculating the needed gradients, but they lack from the needed flexibility. However, both approaches can be combined to a new hybrid method combining their strengths in the following way:

The main drawback of the direct or the adjoint method is the need of analytic partial derivatives of the objective and the constraints with respect to ρ and u. But these derivatives can easily be provided by using AD tools. Then only $\frac{\partial K}{\partial \rho_i}$ and $\frac{\partial F}{\partial \rho_i}$ remain, for which hand-coded routines have to be implemented or AD can be used. For optimal sizing problems, these routines can be hand-coded easily. Furthermore, they do not depend on the specific problem which justifies the additional effort for coding even for more complex problems.

5 Numerical results

In the following, some numerical results for the problem stated in Section 2 are presented. They were calculated on an SGI Origin 2000 with 300 MHz.

At the beginning, we tried to use only few design parameters. Therefore, we divided our domain into a number of sub-domains (see Figure 3) and approximated the thickness with a constant function per sub-domain. The state

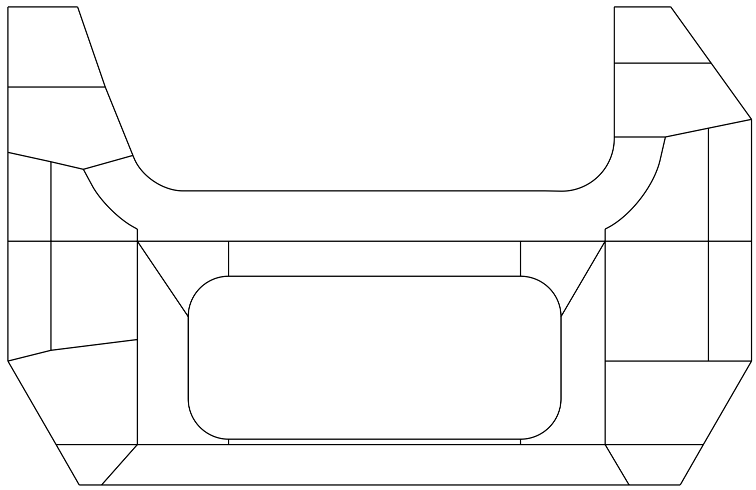

Figure 3: Frame with 24 sub-domains

		Finite Diff.	Pure AD	Hybrid M.
Problem dim.	Nr. of design params	24	24	24
	Nr. of elements (quad.)	3 981	3 981	3 981
	DOFs of state equ.	16 690	16 690	16 690
Optimizer statistics	Iterations	83	100	100
	Function evaluations	19 752	315	236
	Gradient evaluations	84	101	101
Runtime	Total CPU time	12.4 h	4.88 h	0.39 h
	Total elapsed time	12.6 h	8.42 h	0.40 h
Elapsed time	Optimizer	0.01 h	0.03 h	0.01 h
	Function evaluation	0.23 h	2.36 h	0.20 h
	Gradient evaluation	12.40 h	6.00 h	0.18 h

Table 1: Comparison of the runtime for various differentiation strategies

equation was discretized using triangular finite elements with quadratic FE functions.

For evaluating the gradient, either finite differences, a pure AD approach or the hybrid method were used. For a better comparison, the calculation was terminated after a fixed number of steps (The run using finite differences terminated earlier because the search direction was no descent direction anymore). Detailed results can be found in Table 1. All 3 methods lead to a similar design with about 5 % reduction of the mass compared to the starting configuration (which is the current design of the frame). The optimized thickness distribution can be seen in Figure 4, Figure 5 shows the distribution of the van Mises stresses in the optimized frame.

It can be seen in Table 1 that for a few design parameters the main effort consists in solving the FE state equation, respectively calculating the gradient

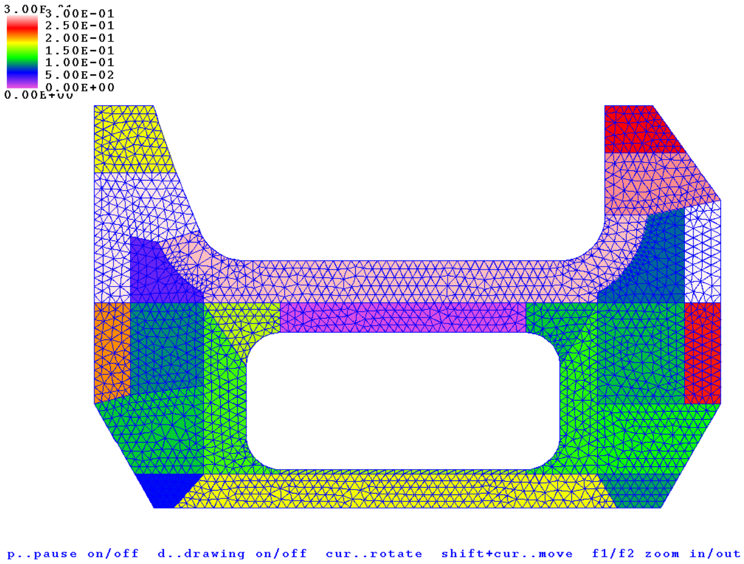

Figure 4: Optimized thickness distribution for 24 sub-domains

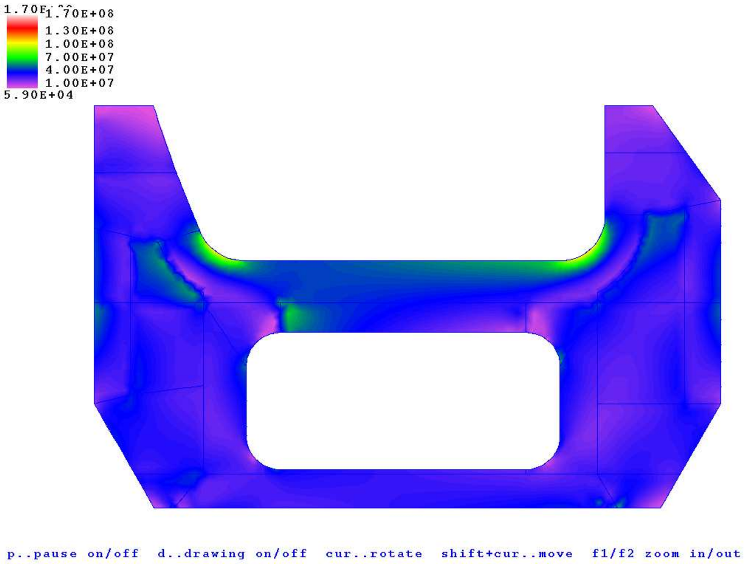

Figure 5: Van Mises stresses for 24 sub-domains

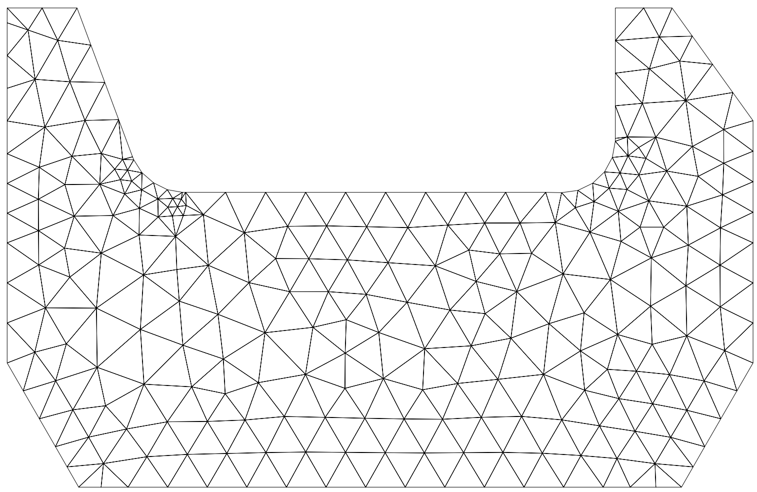

Figure 6: Frame with 449 sub-domains

of the objective. Compared to finite differences and the pure AD approach, the hybrid method is severely faster, as it combines a fast function evaluation and a fast gradient evaluation. The gradient evaluation is the main drawback for finite differences. For the pure AD approach we had to implement additional safeguards. In order to detect when a regeneration of the evaluation graph was necessary, we compared the value of the objective using the evaluation graph and the value using a native C++ implementation which explains the longer runtime of the function evaluation.

In order to find a better suited splitting of the domain in few sub-domains we increased the number of sub-domains to about 450. We used the coarsest grid of our FE triangulation also for discretizing the thickness distribution (c.f. Figure 6). For solving the design problem, each coarse grid finite element was subdivided into 16 elements using 2 levels of uniform refinement. On this refined triangulation the state equation was discretized using finite elements with quadratic FE functions. As finite differences are no more suitable for this number of design parameters, Table 2 contains only results for the pure AD approach and the hybrid method. Additionally, we present also results for a finer discretization of the state equation.

The values reported for the evaluation graph have the following meaning: *Independents* is the number of independent variables of the function differentiated automatically, *dependents* the number of dependent ones (only one dependent variable, as only the objective is differentiated using ADOL-C). *Operations* gives us the number of arithmetic operations in the evaluation graph, *maxlive* the maximal number of live active variables (maximal number of variables allocated at one point of time during the evaluation of the objective), *valstacksize* the size of the value stack (number of intermediate results).

The optimized thickness distribution can be seen in Figure 7, the corresponding stress distribution in Figure 8. Analyzing the runtime behaviour of the two methods in Table 2, it can be seen that the pure AD approach is no

		Pure AD	Hybrid M.	Hybrid M.
Problem dim.	Nr. of design params	449	449	449
	Nr. of elements (quad.)	1 796	1 796	7 184
	DOFs of state equ.	7 518	7 518	29 402
Evaluation graph	Independents	449	9 314	36 586
	Dependents	1	1	1
	Operations	45 521 797	1 399 910	5 578 270
	Maxlive	540 302	28 140	110 122
	Valstacksize	51 995 116	1 644 461	6 552 653
	Total file size	953 MB	32.4 MB	129.2 MB
Optimizer Statistics	Iterations	800	800	800
	Function evaluations	5 811	3 744	3 745
	Gradient evaluations	801	801	801
Runtime	Total CPU time	32.3 h	3.73 h	14.01 h
	Total runtime	38.5 h	3.76 h	14.12 h
Elapsed time	Optimizer	4.0 h	1.93 h	2.64 h
	Function evaluation	16.5 h	1.29 h	8.13 h
	Gradient evaluation	18.0 h	0.54 h	3.35 h

Table 2: Comparison of the runtime for many design parameters

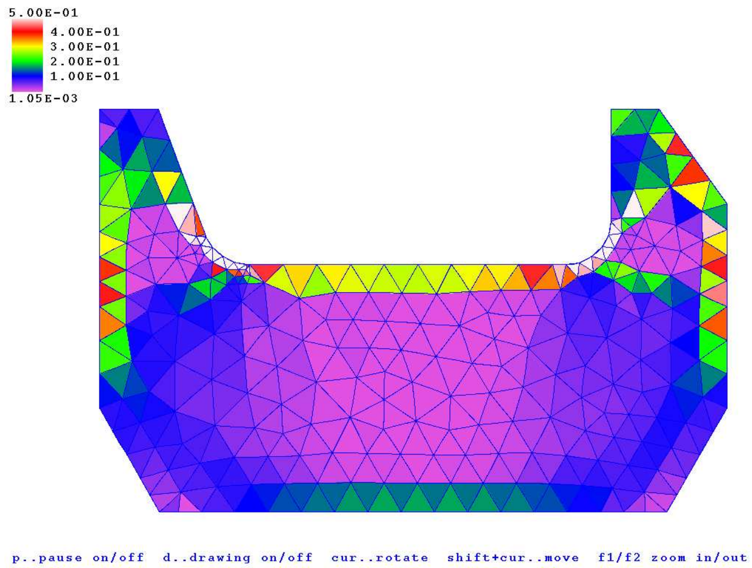

Figure 7: Optimized thickness distribution for 449 sub-domains

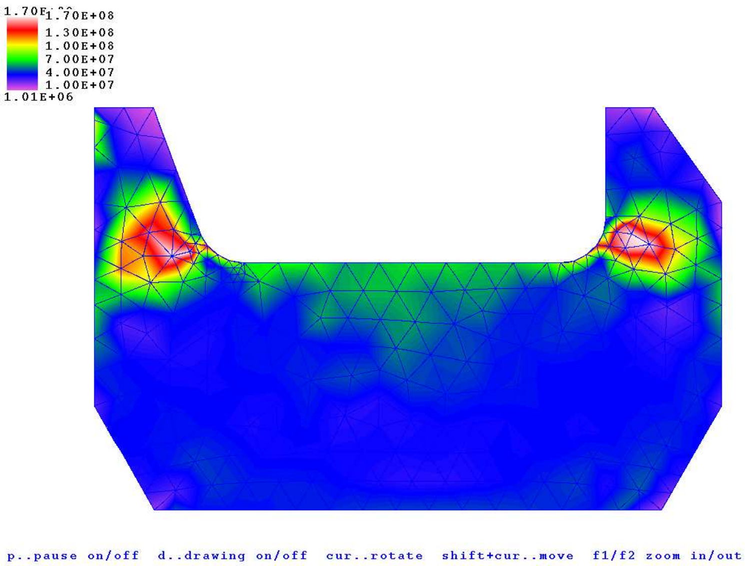

Figure 8: Van Mises stresses for 449 sub-domains

more competitive due to the large file containing the evaluation graph. Furthermore, it can be seen that for the hybrid approach the optimizer needs already a considerable amount of the total runtime. Its relative amount of the runtime even grows when using more design parameters as the complexity of one optimization step is proportional to $(\dim \rho)^3$ (due to the use of dense matrix linear algebra), whereas the complexity of solving one FE state equation is proportional to $\dim u$ (if solvers with optimal complexity e.g. conjugated gradients with multigrid or multilevel preconditioning are used).

Just for comparison, the thickness distribution was discretized using about 1100 design parameters. The optimized thickness distribution and the corresponding distribution of the v. Mises stresses can be found in Figure 9 and Figure 10 respectively. The main problem using that much design parameters is the runtime needed by our optimization module, which dominates the time needed by all function and gradient evaluations completely (about 90 % of the runtime, the FE state equation has about 18000 DOFs).

6 Remarks and Conclusions

In this paper various differentiation strategies needed for solving a real life optimal sizing problem were presented. During the comparison we focussed our attention on the flexibility of the gradient routine and on the possibility to combine the gradient module with fast iterative solvers for the FE state equations. We compared finite differences and the use of automatic differentiation with the adjoint method and the newly developed hybrid approach.

Finite differences provide a flexible way of getting an approximation of the gradient and can be combined easily with iterative solvers for the FE state

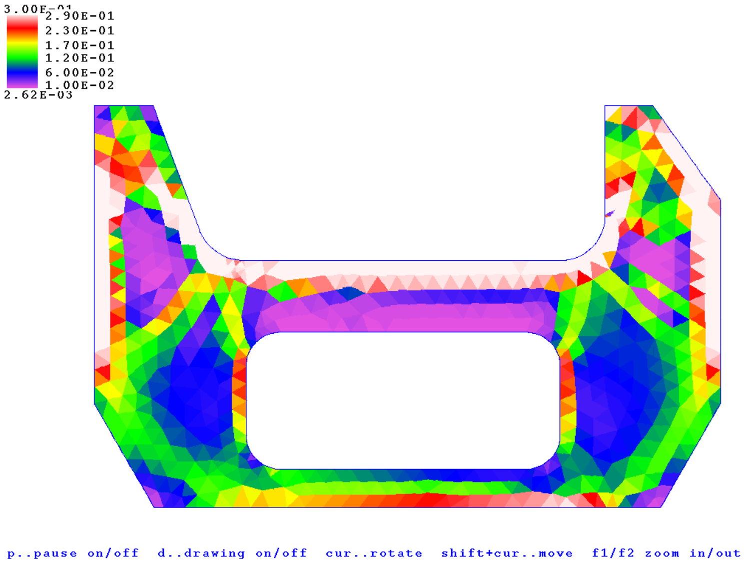

Figure 9: Optimized thickness distribution for 1078 sub-domains

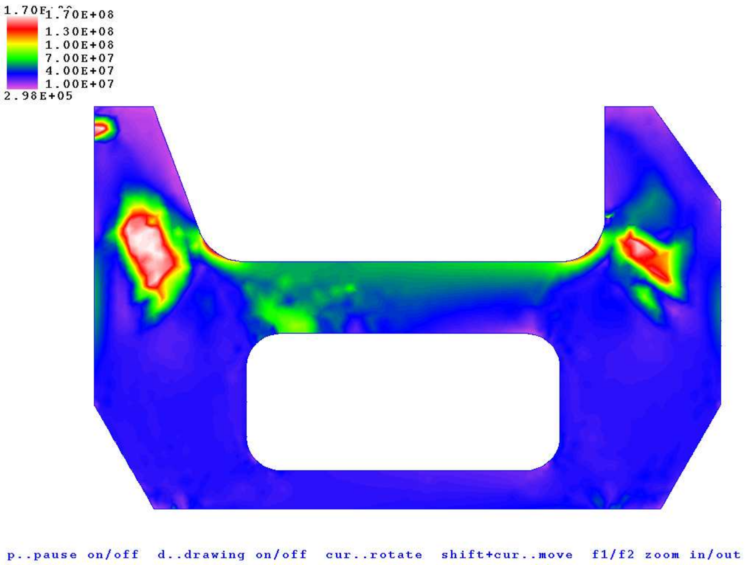

Figure 10: Van Mises stresses for 1078 sub-domains

equations. On the other hand extrapolation schemes have to be used to increase their accuracy, which results in a method with rather high computational effort. Furthermore, finite differences are badly-suited for problems with many design parameters.

Automatic differentiation needs a symbolic evaluation graph and operates on this graph by systematic application of the chain rule. The evaluation time for the gradient is independent of the number of design parameters, but this approach needs huge memory and disk requirements for storing the evaluation graph. Moreover it is not clear how to combine automatic differentiation with iterative solvers for the state equation in a flexible way.

In contrast to finite differences and automatic differentiation, which are black box methods to some extent, the adjoint method is very efficient from the computational point of view if only few constraints are imposed. On the other hand, we loose flexibility, as some parts of the gradient routine have to be re-coded when the objective changes. Moreover, it can easily be combined with iterative solvers for the state equation.

The hybrid method avoids this drawback as it combines the strengths of automatic differentiation and the adjoint method. This method preserves the flexibility of the pure AD approach more or less at the costs of a completely hand-coded gradient routine. Furthermore, the huge memory and disk requirements of the pure AD approach are reduced severely.

Coming back to the optimization routine itself it must be noted that our current implementation of the optimizer is based on dense matrix linear algebra and therefore, it is only well suited for small to medium size optimization problems. But in order to close the gap to topology optimization, which is of high practical importance, new optimization methods for large scale problems have to be developed. An approach using multigrid methods also for solving the optimization problem was proposed by Maar, Schulz [9].

References

[1] M.P. Bendsøe, *Optimization of structural topology, shape and material*, Springer, Berlin, 1995.

[2] C. H. Bischof, A. Carle, P. M. Khademi, and A. Mauer, *The ADIFOR 2.0 system for the automatic differentiation of Fortran 77 programs*, Comput. Sci. Engrg. **3** (1996), no. 3.

[3] R. Flechter, *Practical methods for optimization*, vol. 2, Constrained Optimization, John Wiley & Sons Ltd., Chinchester, 1981.

[4] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*, Academic Press, London-San Diego-New York, 1981.

[5] A. Griewank, *Evaluating derivatives, principles and techniques of algorithmic differentiation*, Frontiers in Applied Mathematics, vol. 19, SIAM, Philadelphia, 2000.

[6] A. Griewank, D. Juedes, and J. Utke, *ADOL-C, a package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Software **22** (1996), no. 3, 131 – 167.

[7] S. P. Han, *A globally convergent method for nonlinear programming*, J. Optimization Theory Appl. **22** (1977), 297 – 309.

[8] J. Haslinger and P. Neittaanmäki, *Finite element approximation for optimal shape design: Theory and applications*, John Wiley & Sons Ltd., Chinchester, 1988.

[9] B. Maar and V. Schulz, *Interior point multigrid methods for topology optimization*, Tech. Report 98-57, IWR, University of Heidelberg, 1998, to appear in Structural Optimization, 2000.

[10] K. G. Mahmoud, *Approximations in optimum structural design*, Advanced in Structural Optimization (B. H. V. Topping and M. Papadrakakis, eds.), Civil-Comp Press, Edinburgh, 1994, pp. 57 – 67.

[11] K. G. Mahmoud, H. Engl, and L. Holzleitner, *Optimum structural design using MSC/NASTRAN and sequential quadratic programming*, Computers & Structures **52** (1994), 437 – 447.

[12] K. Maute and E. Ramm, *Adaptive topology optimization*, Structural Optim. **10** (1995), 100 – 112.

[13] J. Petersson, *Finite element analyses of topology optimization of elastic continua*, ENUMATH 97 - 2nd European Conference on Numerical Mathematics and Advanced Applications (H.G. Bock, G. Kanschat, R. Rannacher, F. Brezzi, R. Glowinski, Y. A. Kuznetsov, and J. Périaux, eds.), World Scientific Publishing, Singapore, 1998, pp. 503 – 510.

[14] M. J. D. Powell, *A fast algorithm for nonlinear constrained optimization calculations*, Numerical Analysis (G. A. Watson, ed.), Lecture Notes in Mathematics, no. 630, Springer, Berlin, 1978.

[15] E. Ramm, K.-U. Bletzinger, R. Reitinger, and K. Maute, *The challenge of structural optimization*, Advanced in Structural Optimization (B. H. V. Topping and M. Papadrakakis, eds.), Civil-Comp Press, Edinburgh, 1994, pp. 27 – 52.

[16] C. Stangl, *Stability of the optimal design with respect to inaccurate solution of the nonlinear state problem*, submitted.

[17] ———, *Optimal sizing for a class of nonlinearly elastic materials*, SIAM J. Optim. **9** (1999), no. 2, 414 – 443.

[18] J. Stoer, *Numerische Mathematik 1*, Springer, Berlin, 1994.