Optimal Sizing and Shape Optimization in Structural Mechanics*

Gundolf Haase Ewald Lindner Wolfram Mühlhuber
Christian Rathberger
Institute of Computational Mathematics
Johannes Kepler University of Linz,
Altenbergerstr. 69, A-4040 Linz, Austria

May 20, 2003

Abstract

This paper is devoted to optimal sizing as well as to shape optimization. As a model example we consider the problem of minimizing the mass of the frame of an injection moulding machine. The deformation of the frame is described by a generalized plane stress state wherein the varying thickness is incorporated in case of the optimal sizing. 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 algorithmic (also called automatic) differentiation of the function evaluation. This approach works fine but requires huge memory and disk capacities. 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 last approach is especially useful in case of a shape optimization. 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.

1 Introduction

The design of a mechanical structure has to fulfill various constraints in many industrial applications. In most cases, an optimal design subject to several constraints is desired. Due to lack of time, engineers designing a machine component have to stop their design process after a few iterations and 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 supporting such a design process have to fulfill mainly two goals. On the one hand they have to be flexible enough to handle the various requirements. Especially, it is desirable to spend only little work when the requirements change. On the other hand, these tools have to be fast.

An extensive review of various methods for structural optimization using finite elements is given in the monograph of HASLINGER, NEITTAANMÄKI [HN88] and in the book of HASLINGER, MÄKINEN [HM03] . A monograph specializing more on

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

topology optimization is BENDSØE [Ben95] and BENDSØE, SIGMUND [BS03]. MAHMOUD [Mah94] focuses on an optimal sizing approach similar to the one in this paper, but he uses approximate representations of the objective and the constraints to reduce the overall costs of the method. STANGL [Sta99] presents a generalization of that approach to a class of nonlinearly elastic materials. 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 [MR95] or RAMM, BLETZINGER, REITINGER, MAUTE [RBRM94].

FORTH, EVANS [FE01] and KIM, HOVLAN [KH01] apply AD to similar problems in fluid dynamics and in the field of inverse problems. TADJOUDDINE, FORTH, PRYCE [TFP01] describe an efficient combination of AD and hand-coded parts of the derivative whose concept is similar to the approach in this paper.

A good introduction into shape optimizaion problems can be found in DELFOUR, ZOLÉSIO [DZ01] and HENROT, SOKOLOWSKI [HS98]. First results of the authors can be found in [HL99, HLLM01].

This paper deals with minimizing the mass of the frame of an injection moulding machine as an example for a typical optimization problem. Since we want to use standard optimization procedures (such as SQP) we devote its main part to the efficient and flexible calculation of the gradients of the given objective and the constraints. We present a very flexible approach using automatic differentiation as well as analytic derivatives inside the code. 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.

We will start with the mathematical description of the optimal sizing problem and we will later emphasize the specifications with respect to the calculation of the gradient in case of a shape optimization.

2 Modeling of the problem

The frame of an injection moulding machine is briefly sketched by its 2D-cut Ω given in Figure 1.

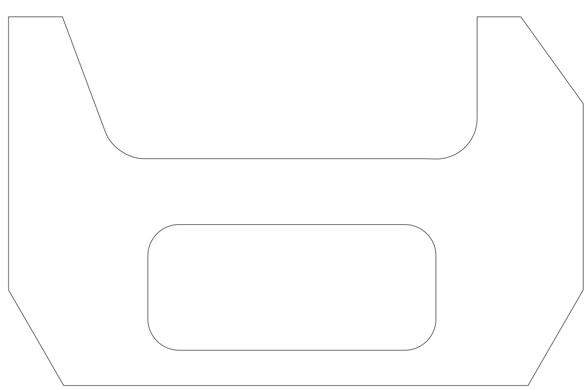

Figure 1: Cross section of the original shape

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
supporting areas	2

The primary goal of the design phase is to minimize the mass of the frame. Several other requirements have to be fulfilled in addition.

Let $V_0 = \{v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma_D, \text{meas } \Gamma_D > 0\}$ denote the set of admissible displacements where $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$. For a fixed thickness $\rho(x)$, the displacement field $u \in V_0$ fulfills the variational equation

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

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 the part Γ_N of the boundary $\partial\Omega$. The design problem can be stated as follows:

$$\begin{array}{rcr} & \displaystyle\int_\Omega \rho \, dx \to \min_{u,\rho} & \\ \text{subject to} & a(\rho; u, v) = F(v) & \text{for all } v \in V_0 \\ & 0 < \underline{\rho} \le \rho \le \overline{\rho}, & \text{a.e. in } \Omega \\ & \sigma^{\text{vM}}(u) \le \sigma^{\text{vM}}_{\max}, \quad \sigma^{\text{ten}}(u) \le \sigma^{\text{ten}}_{\max} & \text{a.e. in } \Omega \\ & \alpha(u) \le \alpha_{\max} & \end{array} \tag{2}$$

$\sigma^{\text{vM}}(u)$ denotes the v. Mises stress, $\sigma^{\text{ten}}(u)$ 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. We denote the discrete approximation of ρ and u again by ρ and u. In our application, the upper limits on the angle and the stresses are treated either 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:

$$\begin{aligned} &\text{mass}(\rho) + \omega_1 \big(\max\,(\|\sigma^{\text{vM}}\|_p - \sigma^{\text{vM}}_{\max}, 0)\big)^2 + \omega_2 \big(\max\,(\|\sigma^{\text{ten}}\|_p - \sigma^{\text{ten}}_{\max}, 0)\big)^2 \\ &\qquad + \omega_3 \big(\max\,(\alpha - \alpha_{\max}, 0)\big)^2 \quad \to \min_{u,\rho} \\ &\qquad \text{subject to} \qquad K(\rho)\, u = F \quad \text{and} \quad \underline{\rho} \le \rho \le \overline{\rho} \;. \end{aligned} \tag{3}$$

3 A short sketch on the optimization strategy

From the optimization's point of view the problem (3) is a special case of

$$\begin{array}{rl} & J(u, \rho) \to \min\limits_{u,\rho} \\ \text{subject to} & K(\rho)\, u = f(\rho) \qquad \text{and} \qquad \underline{\rho} \le \rho \le \overline{\rho} \;, \end{array} \tag{4}$$

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 state equation u is typical for design problems. 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, u can be formally eliminated which leads to

$$\begin{aligned} \tilde{J}(\rho) = J(K^{-1}(\rho)\, f(\rho),\, \rho) \to \min_{\rho} \\ \text{subject to} \qquad \underline{\rho} \leq \rho \leq \overline{\rho} \ . \end{aligned} \tag{5}$$

As we want to use a standard SQP method for optimizing, the formulation in (5) is advantageous compared to (4) as it has much fewer parameters. Details on these kinds of optimization procedures can be found e.g. in GILL, MURRAY, WRIGHT [GMW81] or NOCEDAL, WRIGHT [NW99].

The optimizer used in our code is based on a Quasi-Newton approximation of the Hessian using a modified BFGS update formula following POWELL [Pow78] in order to avoid the need for Hessian information of the objective.

4 Calculating gradients for the optimal sizing problem

Using a Quasi-Newton strategy and update formulas the remaining main problem is the calculation of gradients for the objective and the constraints. In most cases the implementation of analytic derivatives 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. On the one hand we have black box methods like finite differences or automatic differentiation (c.f. GRIEWANK [Gri00]), 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 [HN88]). As none of these methods is really well suited for our problem, a hybrid method combining automatic differentiation and the adjoint method has been developed.

As our finite element code is completely written in C++ and uses heavily virtual inheritance, we use ADOL-C for differentiating our code, c.f. GRIEWANK, JUEDES, UTKE [GJU96]. We applied the reverse mode within our calculations, as we have at most a few nonlinear constraints (c.f. (2), (3)) whereas the dimension of the design space is usually large.

4.1 Direct and adjoint Method

Both methods are well-known in the shape optimization community and take into account the special structure of the FE state equation. Differentiating the discretized state equation with respect to a design parameter ρ_i leads to

$$K\,\frac{\partial u}{\partial \rho_i} = \frac{\partial f}{\partial \rho_i} - \frac{\partial K}{\partial \rho_i}\,u. \tag{6}$$

For the direct method, (6) is solved numerically by some direct or iterative solver. Then the gradient of the objective can be calculated by (c.f. (5))

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\rho_i} = \frac{\partial J}{\partial \rho_i} + \langle \frac{\partial J}{\partial u}, \frac{\partial u}{\partial \rho_i} \rangle. \tag{7}$$

The adjoint method solves (6) formally and inserts the result in (7) which leads to

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\rho_i} = \frac{\partial J}{\partial \rho_i} + \langle K^{-T}\, \frac{\partial J}{\partial u}, \frac{\partial f}{\partial \rho_i} - \frac{\partial K}{\partial \rho_i}\, u\rangle. \tag{8}$$

4.2 Comparison

ADOL-C needs a file containing the evaluation graph of the function in a symbolic form which is generated at runtime. For structural optimization problems, huge memory and disk capabilities are required for that purpose. To give an example, the files storing the evaluation graph for a problem with about 450 design parameters and about 7500 degrees of freedom (DOFs) in the FE state equation take about 1 GB of disk space. The flexibility of ADOL-C with respect to changes in the objective is similar to finite differences. For the 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 [Gri00] and references therein). As the use of iterative methods (e.g. multilevel methods) is important for solving fine discretizations of the state equation efficiently. For the moment, the applicability is limited to problems, where direct solvers can be used.

The direct method needs the solution of one state equation per design parameter, whereas the adjoint method needs the solution of one adjoint problem for the objective and in principle for each constraint. Depending on the number of design parameters and constraints, the better suited method can be chosen. As analytic partial derivatives of J with respect to ρ and u are needed, 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 many more solutions of the FE state problem, compared to AD the huge evaluation graph which originates mainly from the solution of the state equation is avoided. For both methods any solver can be used for solving the state problem, especially the use of iterative solvers like conjugate gradient methods with multilevel preconditioning is recommanded.

4.3 Hybrid Method

Comparing the properties of the direct and the adjoint method and of AD 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 $\frac{\partial K}{\partial p_i}$ and $\frac{\partial f}{\partial p_i}$ 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 of coding even for more complex problems.

		Finite Diff.	Pure AD	Hybrid M.
Problem dim.	Nr. of design par.	24	24	24
	Nr. of elements	3981	3981	3981
	DOFs of state equ.	16690	16690	16690
Optimizer statistics	Iterations	83	100	100
	gradient eval.	12.40 h	6.00 h	0.18 h
	function eval.	0.23 h	2.36 h	0.20 h
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

Table 1: Comparison of the runtime for various differentiation strategies

		Pure AD	Hybrid M.	Hybrid M.
Problem dim.	Nr. of design par.	449	449	449
	Nr. of elements	1796	1796	7184
	DOFs of state equ.	7518	7518	29402
Evaluation graph	Operations	45521797	1399910	5578270
	Total file size	953 MB	32.4 MB	129.2 MB
Optimizer statistics	Iterations	800	800	800
	gradient eval.	18.0 h	0.54 h	3.35 h
	function eval.	16.5 h	1.29 h	8.13 h
Runtime	Total CPU time	32.3 h	3.73 h	14.01 h
	Total elapsed time	38.5 h	3.76 h	14.12 h

Table 2: Comparison of the runtime for many design parameters

5 Numerical results for the optimal sizing problem

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 and approximated the thickness with a constant function per sub-domain. For evaluating the gradient, either finite differences, an AD approach for (5) 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). 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 the following we used the coarsest grid of our FE triangulation for discretizing the thickness. For solving the state problem, each coarse grid element was subdivided into 16 elements using 2 levels of uniform refinement. On this refined triangulation the state equation was discretized. Table 2 contains results for the pure AD approach and the hybrid method. Also results for an even finer discretization of the state equation are presented.

Analyzing the runtime behaviour we see that the pure AD approach is no 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 state equation is proportional to $\dim u$ (if solvers with optimal complexity e.g. conjugate gradients with multigrid or multilevel preconditioning are used).

6 Calculating the gradient for shape optimization

We have seen in the optimal sizing problem that a direct implementation of the gradient can accelerate the code dramatically although it requires more work on the implementation side. We were curious what performance gain can be achieved when the gradient calculation in a 2D shape optimization problem is fully implemented, i.e., no automatic differentiation or finite differences are used therein. The shape under investigation is similar to the one in Fig. 1. This shape can be easily described by corner points (x- and y-coordinates), circular parts of the boundary (x- and y-coordinates of the center plus the radius) connected with straight lines, see Fig. 2. Our set of design parameters P

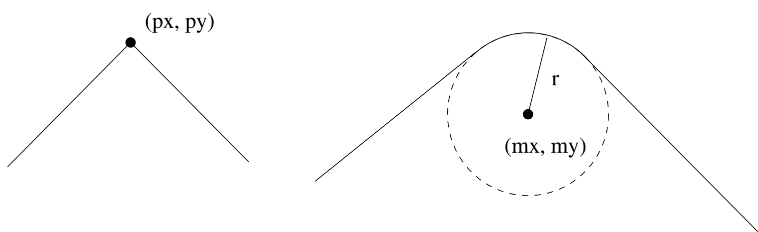

Figure 2: Possible usage of design parameters in Shape Optimization

contains all these parameters p_x, p_y, m_x, m_y, r which are restricted via box constraints. More details on topics discussed in the following sections can be found in the master thesis by C. Rathberger [Rat02].

6.1 A second look at the gradient

If we pick an arbitrary design parameter $p \in P$ and if we assume that our objective J depends only on the mass, the displacement in certain points and the resulting van Mises stress σ^{vM} (the handling of tensile stresses will be similar) we can write:

$$J = J(p, u(p), \sigma^{vM}(p, u(p))) \tag{9}$$

where the explicit dependency on p is due to the fact that for homogeneous objects the mass depends only on the boundary shape. If we want to calculate the total differential with respect to design parameter p this leads to

$$\begin{aligned} \frac{dJ}{dp} &= \frac{\partial J}{\partial p} + \frac{\partial J}{\partial u} \cdot \frac{du}{dp} + \frac{\partial J}{\partial \sigma^{vM}} \cdot \left(\frac{\partial \sigma^{vM}}{\partial p} + \frac{\partial \sigma^{vM}}{\partial u} \cdot \frac{du}{dp} \right) \\ &= \frac{\partial J}{\partial p} + \frac{\partial J}{\partial \sigma^{vM}} \cdot \frac{\partial \sigma^{vM}}{\partial p} + \left(\frac{\partial J}{\partial u} + \frac{\partial J}{\partial \sigma^{vM}} \cdot \frac{\partial \sigma^{vM}}{\partial u} \right) \cdot \frac{du}{dp} \end{aligned} \tag{10}$$

Again, we want to eliminate the term $\frac{du}{dp}$ by differentiating the state equation $Ku = f$ with respect to the design variable p and we get similarly to (6)

$$\frac{du}{dp} = K^{-1}\left(\frac{df}{dp} - \frac{dK}{dp}\cdot u\right) \tag{11}$$

Inserting equation (11) into equation (10) results in

$$\begin{aligned}
\frac{dJ}{dp} &= \frac{\partial J}{\partial p} + \frac{\partial J}{\partial \sigma^{vM}}\cdot\frac{\partial \sigma^{vM}}{\partial p} + \\
&\quad + \left(\frac{\partial J}{\partial u} + \frac{\partial J}{\partial \sigma^{vM}}\cdot\frac{\partial \sigma^{vM}}{\partial u}\right)\cdot K^{-1}\left(\frac{df}{dp} - \frac{dK}{dp}\cdot u\right) \\
&= \underbrace{\frac{\partial J}{\partial p}}_{(i)} + \underbrace{\frac{\partial J}{\partial \sigma^{vM}}\cdot\frac{\partial \sigma^{vM}}{\partial p}}_{(ii)} + \\
&\quad + \underbrace{\left\langle K^{-1}\cdot\left(\frac{\partial J}{\partial u} + \frac{\partial J}{\partial \sigma^{vM}}\cdot\frac{\partial \sigma^{vM}}{\partial u}\right), \frac{df}{dp} - \frac{dK}{dp}\cdot u\right\rangle}_{(iii)}
\end{aligned} \tag{12}$$

where we have used the fact that K is symmetric in the last transformation. We will investigate the three principal parts of the derivative in (12) separately:

(i) As the objective only depends explicitly on the design parameters through the mass, this partial derivative can be evaluated as

$$\frac{\partial J}{\partial p} = \varrho\cdot\frac{\partial V}{\partial p} = \varrho\cdot d\cdot\frac{\partial A}{\partial p} = \varrho\cdot d\cdot\sum_{e=1}^{m}\frac{\partial A_e}{\partial p} \tag{13}$$

where V is the volume, A the area, A_e the area of element e, ϱ the density of the material and d the thickness of the frame. Writing this differential as sum over all elements has the advantage that the evaluation is independent from the geometrical properties of the boundary shape and it is not even inefficient regarding the computational effort. The differential of the element areas by the design parameters will be required in equation (18) in Section 6.2 anyway.

(ii) Differentiating the objective by the van Mises stress gives us only some constant of proportionality. Partially differentiating the van Mises stress by the design parameters is a little more complicated, but can be basically reduced to repetitive applications of product and chain rule differentiation. We will give the explicit expressions in Section 6.4.

(iii) Finally, the handling of the derivatives in the scalar product $< .,. >$ is the main effort and requires some more considerations.

Examining the left hand side of the scalar product in equation (12) in more detail, the first thing we can do is to rewrite it as the adjoint problem

$$\begin{aligned}
v_{left} &:= K^{-1}\cdot\left(\frac{\partial J}{\partial u} + \frac{\partial J}{\partial \sigma^{vM}}\cdot\frac{\partial \sigma^{vM}}{\partial u}\right) \\
K\cdot v_{left} &= \frac{\partial J}{\partial u} + \frac{\partial J}{\partial \sigma^{vM}}\cdot\frac{\partial \sigma^{vM}}{\partial u},
\end{aligned} \tag{14}$$

where K is the stiffness matrix already known from the state problem. We obviously can evaluate the left hand side v_{left} simply by solving the state problem with the only difference that we use a different right hand side instead of f, namely

$$\frac{\partial J}{\partial u} + \frac{\partial J}{\partial \sigma^{vM}} \cdot \frac{\partial \sigma^{vM}}{\partial u} \tag{15}$$

Here the partial differential of the objective by the displacement field u will be an n-dimensional null vector except for those entries corresponding to nodes on the boundary affected by angular constraints. The differentials involving the van Mises stress again will considered in Section 6.4.

6.2 Differentiating the state equation

First we have a closer look at the right hand side $\frac{df}{dp} - \frac{dK}{dp} \cdot u$ of the scalar product (iii) in (12). The gravity of mass can be neglected in comparison to the large forces acting on the C-frame. Additionally, the shape is also strictly fixed in those regions where these surface tractions act. Therefore, we can use $\frac{\partial f}{\partial p} = 0$ in our considerations (see [Rat02, §10.1]). The displacement field u is already available from solving the state equations $Ku = f$, The differential of the stiffness matrix K can be expressed by the differentials of the element stiffness matrices K_e through the connectivity matrices C_e.

$$\frac{dK}{dp} = \sum_{e=1}^{m} C_e \frac{dK_e}{dp} C_e^T \tag{16}$$

We took advantage of the fact that the connectivity matrices contain only "topological" data and they are therefore independent from the design parameters. Additionally, we restrict ourselves to triangular finite elements with quadratic shape functions. The derivative of the element stiffness matrix will be shown for one of the 4×36 entries of K_e (the others work completely by analogy). Selecting K_{xy}^{12} from K_e, i.e., the discretization of $\frac{\partial \psi^{(1)}}{\partial x} \cdot \frac{\partial \psi^{(2)}}{\partial y}$ with $\psi^{(1)}$ and $\psi^{(2)}$ denoting two quadratic shape functions of the triangular finite element, we get

$$K_{xy}^{12} = \frac{1}{4A_e}[\nu(x_1-x_3)(y_2-y_3) + \frac{(1-\nu)}{2}(y_3-y_1)(x_3-x_2)] \tag{17}$$

and consequently

$$\begin{aligned} \frac{dK_{xy}^{12}}{dp} &= -\frac{1}{4A_e^2}\frac{dA_e}{dp}[\nu(x_1-x_3)(y_2-y_3) + \frac{(1-\nu)}{2}(y_3-y_1)(x_3-x_2)] \\ &\quad + \frac{1}{4A_e}[\nu\{(\frac{dx_1}{dp}-\frac{dx_3}{dp})(y_2-y_3) + (x_1-x_3)(\frac{dy_2}{dp}-\frac{dy_3}{dp})\} \\ &\quad + \frac{(1-\nu)}{2}\{(\frac{dy_3}{dp}-\frac{dy_1}{dp})(x_3-x_2) + (y_3-y_1)(\frac{dx_3}{dp}-\frac{dx_2}{dp})\}] \end{aligned} \tag{18}$$

which looks very complicated, but again consists only of differentials of A_e and the vertices of the element by the design parameter p.

The element area A_e can be easily calculated for triangular elements (with corner points (x_1, y_1),(x_2, y_2) and (x_3, y_3)) and the appropriate derivative is

$$\begin{aligned} \frac{dA_e}{dp} &= \frac{1}{2} \cdot [(\frac{dx_2}{dp} - \frac{dx_1}{dp})(y_3 - y_1) + (x_2 - x_1)(\frac{dy_3}{dp} - \frac{dy_1}{dp}) \\ &\quad -(\frac{dx_3}{dp} - \frac{dx_1}{dp})(y_2 - y_1) - (x_3 - x_1)(\frac{dy_2}{dp} - \frac{dy_1}{dp})] \end{aligned} \tag{19}$$

So the differential of the element area can be expressed by the differential of the corner points by the design parameters as well. Obviously the central task when trying to differentiate the objective is differentiating the coordinates of the nodes by the design parameters.

Therefore, the difficulties in the gradient calculation are shifted to the dependencies of the finite element mesh from the changes in the geometry. That dependency consists again of two parts:

$$\text{Design parameters } P \xrightarrow{i.} \text{ Boundary Nodes } \Gamma_h \xrightarrow{ii.} \text{ Interior Nodes } \Omega_h$$

i. The boundary mapping from P to Γ_h is described in detail in [Rat02, §9.1]. Differentiating this expression will be extremely long-winded though mathematically not very difficult [Rat02, §10.3] and we skip this part in the paper.

ii. The mesh mapping from the boundary Γ_h to the interior nodes Ω_h requires a transformation of the f.e. mesh. For calculating this differential we will have to differentiate the mesh smoother, which is either an algorithm based on the Jacobi method or one involving the solution of the linear elasticity problem.

If we take into consideration the way the mesh coordinates are stored in our data structure (coordinates of the initial geometry plus a certain deformation field) we can write the mesh deformation with respect to the design parameters more precisely:

$$x_i = x_i^{initial} + (\underline{u}_m)_i(\Gamma(P)) \quad \forall i \in \omega_h \tag{20}$$

where $\underline{u}_m$ is the result of the mesh smoothing problem. Consequently this leads to

$$\frac{dx_i}{dp} = \sum_{j \in \gamma_h} \frac{dx_i}{dx_j} \cdot \frac{dx_j}{dp} \quad \forall i \in \omega_h \tag{21}$$

6.3 Differentiating the mesh mapping

Talking about the mesh mapping we have to distinguish between the two possible implementations. Let us first discuss the very easy case of the Jacobi iteration.

6.3.1 Differentiating the Jacobi iteration

If we want to calculate the derivative of the Jacobi method we have to differentiate the iteration scheme

$$x_j^{t+1} = \frac{1}{|N_j|} \sum_{i \in N_j} x_i^t \quad \forall j \in \omega_h \tag{22}$$

which it is based on. Where we first had to transform the boundary nodes (not affected by the smoothing algorithm) and then iterate according to equation (22) to extend the transformation to the interior, we now first have to differentiate the boundary nodes by the desired design parameter and then extend the derivative into the interior by using the differentiated iteration scheme

$$\frac{dx_j^{t+1}}{dp} = \frac{1}{|N_j|} \sum_{i \in N_j} \frac{dx_i^t}{dp} \quad \forall j \in \omega_h \tag{23}$$

The idea behind the derivative is precisely the same as for the mesh mapping itself.

As brilliantly simple as this may seem it is also a grave disadvantage, because this means that we have to repeat the (already very high) computational effort required for mesh mapping again for each design parameter in order to get the gradient. Of course the number of iterations has to be the same for the mapping as for the derivative in order to get correct results! So if the Jacobi iteration was for a certain problem 5 times slower than the elasticity mapping and we have 9 design parameters, this widens the gap to a factor $(1+9)\cdot 5 = 50$ at least! That would be for example more than a day compared to 30 minutes!

Although we could reduce the computational effort a little by calculating the differential of the mesh smoother in the same loop as the mesh smoother itself, this does not change the general disadvantages of this approach. Obviously we should *really* find a different smoother!

6.3.2 Differentiating the elastic mesh mapping

Looking a little bit closer at how the deformation field for the mesh $\underline{u}_m$ is influenced by the deformation of the boundary Γ we recall that it is calculated as solution of

$$\begin{aligned} &K_h \cdot \underline{u}_w = 0 \\ &\text{with Dirichlet boundary conditions dependent on } \Gamma \quad , \end{aligned} \tag{24}$$

where K_h itself is independent from the design parameters, because the deformed geometry is always calculated from the optimal, undeformed initial geometry and therefore the K_h for mesh transformation is always (the same) stiffness matrix for the initial setting of the design parameters. So we have to differentiate the solution of the given system of linear equations by the Dirichlet boundary conditions. How can this be done?

For answering this question we have to remind ourselves how Dirichlet data is actually incorporated into the linear system. One way to do this is by discrete homogenization, which leads to (if we look at only one equation, respectively row, of the system)

$$\left(K_{inner} \cdot \underline{u}_m^{inner}\right)_i = \left(-\sum_{k\in\gamma_h} K_{ik}^h \cdot x_k\right)_i \qquad \forall i \in \omega_h \tag{25}$$

Solving for $\underline{u}_m$ and differentiating by the coordinate of a boundary node (this is what we actually have to do accordingly to equation (21)) leads to

$$\frac{d\underline{u}_m}{dx_j} = -K_{inner}^{-1} \cdot \hat{K}_{\bullet j} \tag{26}$$

where $\hat{K}_{\bullet j}$ is the j-th column of the initial Matrix K_h with the lines corresponding to the boundary nodes removed. Equation (26) results from

$$\frac{dx_k}{dx_j} = \delta_{jk} \tag{27}$$

and it requires to solve one linear system per boundary node! This would be, of course, terribly ineffective and is therefore absolutely unthinkable! Furthermore storing the results of equation (26) would require an array of the dimensions $|\omega_h| \times |\gamma|$, which is far too large to be practicable as well. But both problems can be solved with one idea.

If we now calculate $\frac{d\underline{u}_m}{dp}$ instead of $\frac{d\underline{u}_m}{dx_j}$, then equation (20) yields the equivalence

$$\frac{d\underline{u}_m}{dp} = \frac{dx}{dp}, \tag{28}$$

because $x^{initial}$ is constant in this context. Using equation (23) this leads to

$$\begin{aligned}\frac{dx}{dp} &= \left(\sum_{j\in\gamma_h}\frac{dx_i}{dx_j}\cdot\frac{dx_j}{dp}\right)_{i\in\overline{\omega}_h} = \left(\left\{\begin{array}{ll}\sum_{j\in\gamma_h}\frac{dx_i}{dx_j}\cdot\frac{dx_j}{dp} & i\in\omega_h\\ \frac{dx_i}{dp} & i\in\gamma_h\end{array}\right.\right)_{i\in\overline{\omega}_h}\\ &= \left(\left\{\begin{array}{ll}-\sum_{j\in\gamma_h}\left(K_{inner}^{-1}\cdot\hat{K}_{\bullet j}\right)_i\cdot\frac{dx_j}{dp} & i\in\omega_h\\ \frac{dx_i}{dp} & i\in\gamma_h\end{array}\right.\right)_{i\in\overline{\omega}_h}\end{aligned}$$

which can be rewritten as

$$\frac{dx}{dp} = M^{-1}\cdot\sum_{j\in\gamma_h}N_{\bullet j}\cdot\frac{dx_j}{dp} \tag{29}$$

where

$$\begin{aligned}M &= (-K \text{ with boundary-rows and -columns set to } \delta_{ij})\\ N &= (K \text{ with boundary-rows set to } \delta_{ij})\\ N_{\bullet j} &= \text{j-th column of } N.\end{aligned}$$

Furthermore setting

$$b = \sum_{j\in\gamma_h}N_{\bullet j}\cdot\frac{dx_j}{dp} \tag{30}$$

leads to

$$M\cdot\frac{dx}{dp} = b \tag{31}$$

so that we now can evaluate equation (21) with the effort of solving only one linear system (solved with PEBBLES[1], see also [Rei01]) per design parameter. We furthermore notice from (30) that only b depends on the boundary and therefore on the design parameters in equation (31). The matrix M has to be assembled only *once*, which speeds things up even further.

6.4 Differentiating the van Mises stress

What now remains in term (iii) and (ii) of equation (12) are the two derivatives

$$\frac{\partial\sigma^{vM}}{\partial u} \text{ and } \frac{\partial\sigma^{vM}}{\partial p} \quad , \tag{32}$$

where the second can be written as

$$\frac{\partial\sigma^{vM}}{\partial p} = \frac{\partial\sigma^{vM}}{\partial x}\cdot\frac{\partial x}{\partial p} \quad . \tag{33}$$

To evaluate this we remind that the van Mises stress is calculated by interpolating the displacement field u on an element e. We split up the mapping into three parts, where a_x, b_x, a_y, b_y denote the derivatives of the f.e. shape function (expressed as $a\cdot x+b\cdot y+c$) and gain

$$\begin{aligned}(\vec{x},\vec{u}) &\longrightarrow (a_x,b_x,a_y,b_y)\\ (a_x,b_x,a_y,b_y) &\longrightarrow (\sigma_{11},\sigma_{22},\sigma_{33},\sigma_{12})\\ (\sigma_{11},\sigma_{22},\sigma_{33},\sigma_{12}) &\longrightarrow \sigma^{vM}\end{aligned} \tag{34}$$

[1] http://www.numa.uni-linz.ac.at/Research/Projects/pebbles.html

How complicated this gets can be seen by calculating

$$\frac{\partial \sigma^{vM}}{\partial u} = \frac{\partial \sigma^{vM}}{\partial \sigma_{11}} \left(\frac{\partial \sigma_{11}}{\partial a_x} \frac{\partial a_x}{\partial u} + \frac{\partial \sigma_{11}}{\partial a_y} \frac{\partial a_y}{\partial u} + \frac{\partial \sigma_{11}}{\partial b_x} \frac{\partial b_x}{\partial u} + \frac{\partial \sigma_{11}}{\partial b_y} \frac{\partial b_y}{\partial u} \right) + \frac{\partial \sigma^{vM}}{\partial \sigma_{22}} \quad \ldots$$

As the principle behind all this is always the same and fairly easy, we will not go into further detail here.

6.5 Implementation

Summing up the results of the previous sections we now can list the sequence of program steps required for evaluating the gradient at a given point in the parameter space. We will also keep track of the number of linear systems (LIN) that have to be solved. Evaluation of equation (12) takes place in the following way:

1. Solve the state problem and store the displacement field u. (1 LIN)
2. Calculate and store $\frac{dx_j}{dp}$, $j \in \gamma_h$ for all boundary nodes and design parameters. This is the derivative of the boundary mapping.
3. Calculate (using the results from the previous step) and store $\frac{dx_i}{dp}$, $i \in \overline{\omega}_h$ for all nodes and all design parameters. This is the derivative of the mesh mapping. ($|P|$ LIN)
4. Calculate $\frac{dA_e}{dp}$ for all elements and all design parameters.
5. Calculate the left hand side (LHS) of

$$\left\langle \underbrace{K^{-1} \cdot \left(\frac{\partial J}{\partial u} + \frac{\partial J}{\partial V} \cdot \frac{\partial V}{\partial u} \right)}_{LHS}, \underbrace{\frac{df}{dp} - \frac{dK}{dp} \cdot u}_{RHS} \right\rangle \tag{35}$$

   using the stiffness matrix from the state problem in step 1. and store it. (1 LIN)

6. Calculate $\frac{df}{dp}$ and initialize the RHS of the scalar product with the result.
7. Calculate $\frac{dK}{dp}$ by means of element matrices and update the RHS element-wise with $-\frac{dK_e}{dp} u_e$.
8. Evaluate the scalar product $< LHS, RHS >$
9. Evaluate the terms of equation (12) outside the scalar product.
10. Sum up the results and store them into the respective row of the gradient

The steps 6. to 10. will have to be repeated for each design parameter. As a whole the number of linear systems to be solved is obviously the number of design parameters plus two with the additional aspect that several of the field problems share the same stiffness matrix and differ only in the load vector. If used correctly, this increases the speed of the algorithm even more.

A further (and completely different) possibility for calculating the gradient is the automatic differentiation approach presented in Griewank [Gri00].

7 Numerical results for the shape optimization

The formulation for the design problem was already introduced in Section 2 with the only difference that now the geometry Ω changes but the thickness ρ remains constant. We do not take into account the hole in the middle of the C-frame because we want to simplify the geometry. We use a mesh with 828 triangular finite elements. If we now start the shape optimization with all the constraints from Section 2, we see that the critical constraints will be the angles of the wings. Although σ_{11} can reach critical values on some single elements, on most elements the constraints on the stresses are automatically fulfilled if the constraints on the angles are fulfilled. In the optimal design both wings almost reach their maximal allowed deformation, which can be seen in figure 3. The final design fulfills all constraints and the mass has been reduced to 81.83% of

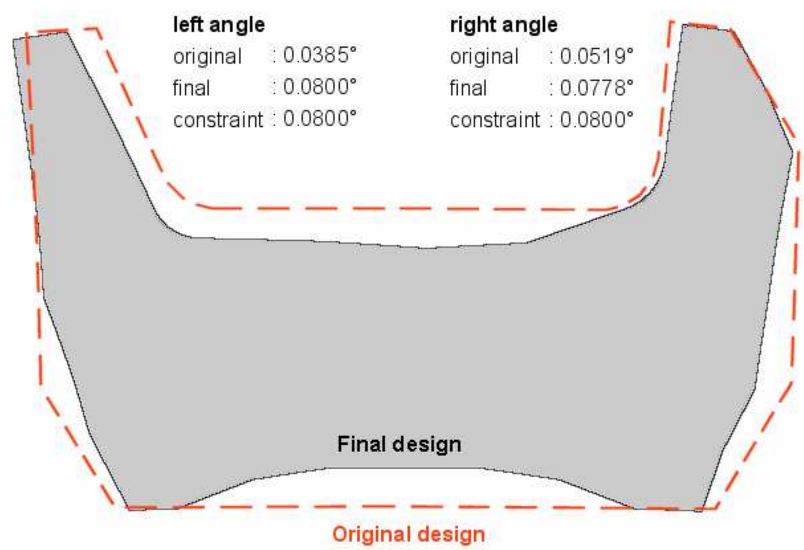

Figure 3: Comparison of the original and of the final deformed geometry

the original value. For the original mass of 5.4223t that means a weight reduction of 985.1kg. The optimization process required 79 iterations and 43.2 seconds for a set of 29 design parameters. That is dramatic reduction in computing time compared with approximately 6 hours when finite differences were used in the gradient calculation.

8 Remarks and conclusions

In this paper differentiation strategies needed for solving a real life design 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 iterative solvers for the state equation. A hybrid method combining the strengths of AD and the adjoint method was presented. 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 reduces severely. The fully implemented gradient in the shape optimization approach shows clearly the very high acceleration that can be achieved. Nevertheless that implementation depends strongly on the geometrical description of the boundary and on the specifications of the finite elements chosen for discretization. Therefore this approach is very fast but not as flexible.

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 [MS00].

References

[Ben95] M.P. Bendsøe. *Optimization of Structural Topology, Shape and Material*. Springer, Berlin, 1995.

[BS03] M.P. Bendsøe and O. Sigmund. *Topology Optimization*. Theory, Methods, Applications. Springer, Berlin, 2nd edition, 2003.

[CFG+01] G.F. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors. *Automatic Differentiation: from Simulation to Optimization*, New York, 2001. Springer.

[DZ01] M. C. Delfour and J.-P. Zolésio. *Shapes and Geometries: analysis, differential calculus, and optimization*. Advances in Design and Control. SIAM, Philadelphia, 2001.

[FE01] S. A. Forth and T. Evans. Aerofoil optimisation via automatic differentiation of a multigridcell-vertex euler flow solver. In *[CFG+01], chapter 13*, 2001.

[GJU96] 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(3):131 – 167, 1996.

[GMW81] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London-San Diego-New York, 1981.

[Gri00] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*, volume 19 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 2000.

[HL99] G. Haase and E. H. Lindner. Advanced solving techniques in optimization of machine components. *Computer Assisted Mechanics and Engineering Sciences*, 6(3):337–343, 1999.

[HLLM01] Gundolf Haase, Ulrich Langer, Ewald Lindner, and Wolfram Mühlhuber. Optimal sizing using automatic differentiation. In K.-H. Hoffmann, R.H.W. Hoppe, and V. Schulz, editors, *Proceedings of "Fast Solution of Discretized Optimization Problems", Berlin*, ISNM 138, 120–138, 2001. Birkhäuser.

[HM03] J. Haslinger and R. Mäkinen. *Introduction to Shape Optimization: Theory, Approximation and Computation*, volume 7 of *Advances in Design and Control*. SIAM, Philadelphia, 2003.

[HN88] J. Haslinger and P. Neittaanmäki. *Finite Element Approximation for Optimal Shape Design: Theory and Applications*. John Wiley & Sons Ltd., Chinchester, 1988.

[HS98] A. Henrot and J. Sokolowski. *A shape optimization problem for the heat equation, Applied Optimization; Optimal Control: Theory, Algorithms, and Applications*. Kluwer Academic Publishers B.V.,, 1998.

[KH01] J. G. Kim and P. D. Hovland. Sensitivity analysis and parameter tuning of a sea ice model. In *[CFG+01], chapter 5*, 2001.

[Mah94] K. G. Mahmoud. Approximations in optimum structural design. In B. H. V. Topping and M. Papadrakakis, editors, *Advanced in Structural Optimization*, pages 57 – 67. Civil-Comp Press, Edinburgh, 1994.

[MR95] K. Maute and E. Ramm. Adaptive topology optimization. *Structural Optim.*, 10:100 – 112, 1995.

[MS00] B. Maar and V. Schulz. Interior point multigrid methods for topology optimization. *Structural Optimization*, 19(3):212–224, 2000.

[NW99] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Reasearch. Springer, New York, 1999.

[Pow78] M. J. D. Powell. A fast algorithm for nonlinear constrained optimization calculations. In G. A. Watson, editor, *Numerical Analysis*, number 630 in Lecture Notes in Mathematics. Springer, Berlin, 1978.

[Rat02] C. Rathberger. Fast product design with modern optimization software – shape optimization[2]. Master's thesis, Johannes Kepler University of Linz, Linz, July 2002.

[RBRM94] E. Ramm, K.-U. Bletzinger, R. Reitinger, and K. Maute. The challenge of structural optimization. In B. H. V. Topping and M. Papadrakakis, editors, *Advanced in Structural Optimization*, pages 27 – 52. Civil-Comp Press, Edinburgh, 1994.

[Rei01] S. Reitzinger. *Algebraic Multigrid Methods for Large Scale Finite Element Equations*. Number 36 in Schriften der Johannes-Kepler-Universität Linz, Reihe C - Technik und Naturwissenschaften. Universitätsverlag Rudolf Trauner, 2001.

[Sta99] C. Stangl. Optimal sizing for a class of nonlinearly elastic materials. *SIAM J. Optim.*, 9(2):414 – 443, 1999.

[TFP01] M. Tadjouddine, S. A. Forth, and J. D. Pryce. AD tools and prospects for optimal AD in CFD flux Jacobian calculations. In *[CFG+01], chapter 27*, 2001.

[2]http://www.numa.uni-linz.ac.at/Staff/haase/Thesis/rathberger-diplom.pdf