G. Allaire · O. Pantz

Structural optimization with FreeFem++

Received: 23 June 2005 / Revised manuscript received: 7 September 2005 / Published online: 8 July 2006
© Springer-Verlag 2006

Abstract The aim of this paper is to show that relatively small, simple, and efficient shape optimization routines can be written using the free finite element software FreeFem++. This is illustrated by the implementation of two classical methods: the boundary variation method and the homogenization one. Even though these routines are simple enough so that their implementation can be assigned (partially or totally) as homework to graduate students, they yield results accurate enough to be useful tools for engineers or researchers.

Keywords Shape and topology optimization · Finite element software

1 Introduction

It is worthless to emphasize that a course on structural optimization must be illustrated by several examples of the methods introduced. It is better to let the students use structural optimization softwares or (even better) to let them write their own. Nevertheless, it is not realistic to ask a student to implement all parts of such algorithms as mesh generation or the finite elements method (for unstructured mesh). On the other hand, using a well-established finite element software, the difficulty is usually moved to the optimization loop (including the adjoint analysis). The goal of this paper is to exhibit (at least) one good compromise between easy optimization and easy finite element analysis, which makes it possible for graduate students to develop their own structural optimization programs. Indeed, as shown in the following, relatively small (about 300 lines) and simple structural optimization routines can be written using FreeFem++, a free and user friendly 2-d finite element software (Hecht et al. 2005). Depending on the available time, it is possible to give the students most of the script and to let them complete some missing parts, as the variational formulations associated to the state and adjoint equations. Our experience is based on a graduate course taught at Ecole Polytechnique (Allaire 2006).

We have implemented two classical structural optimization methods: Hadamard method for geometric optimization (Pironneau 1984; Sokolowski and Zolesio 1992) and the homogenization method for topology optimization (Allaire 2001; Bendsoe and Sigmund 2003). The paper is divided in two independent parts, each one being devoted to one method. For each part, after recalling the principles of the method (we refer to the above quoted textbooks for a more complete presentation), we describe briefly its implementation in FreeFem++. The free software FreeFem++ requires as input a script that describes the geometry of the mesh, the variational formulation of the problem, and the optimization loop. These ingredients have to be provided by the user, all other aspects of finite elements being automatically managed by FreeFem++, including mesh generation, adaptation and deformation, assembling the rigidity matrix, solving the linear system, displaying the result, etc. This is different in spirit from other softwares like Matlab (see Sigmund 2001 for an application in structural optimization). Our routines (i.e., FreeFem++ scripts, tested with the 1.47 version) are freely available on the web page http://www.cmap.polytechnique.fr/~optopo so that anybody can reproduce our illustrative numerical examples (and get the inspiration to create new ones).

2 Boundary variations or geometric optimization

2.1 The gradient method

The gradient method is probably the simplest tool of optimization but it may become tricky when applied to shapes, so we indulge ourselves in giving some details. Let \( F \) be a map from a Hilbert space \( X \) into \( \mathbb{R} \). The gradient method amounts to build a sequence of elements \((x_n)_{n \geq 0} \in X\) by

\[
x_{n+1} = x_n - h_n d_n,
\]

where \( h_n \in \mathbb{R}^+ \) is a small positive step and \( d_n \) is the descent direction defined by

\[
(d_n, y)_X = \langle F'(x_n), y \rangle_{X^*, X} \text{ for any } y \in X.
\]
Usually, the identification between $X$ and its dual $X^*$ under the scalar product $(.,.)_X$ is understood. If we do so, $d_n$ is nothing more but the gradient of $F$, $F(x_n)$. If $F'(x_n)$ is not equal to zero, then for $h_n$ to be small enough, $F(x_{n+1}) < F(x_n)$. The algorithm is initialized with any element $x_0 \in X$, and if $F$ is strongly convex, $x_n$ converges toward the optimal solution $x_*$ of the problem

$$F(x_*) = \min_{x \in X} F(x).$$

In structural optimization, the search set $X$ is no more a Hilbert space but a subset of the open sets of $\mathbb{R}^N$ (where usually, $N = 2$ or $3$) and has neither straightforward differentiable nor Hilbert structure. Nevertheless, the gradient method can successfully be applied. To this end, we need to define variations of open sets and endow the set of variations with a Hilbert (or at least Banach) structure.

### 2.2 Variations of an open set

Let $J(\Omega)$ be a real valued function defined for any open set $\Omega$ of $\mathbb{R}^N$. Let $\Omega$ be a regular open set of $\mathbb{R}^N$. Given a map $\theta$ from $\Omega$ into $\mathbb{R}^N$, we set

$$\Omega(\theta) = (\text{Id} + \theta)(\Omega) \equiv \{ x + \theta(x) \text{ s.t. } x \in \Omega \}.$$

For small vector field $\theta$, the open set $\Omega(\theta)$ are one-to-one perturbations of the initial set $\Omega$. If the map $F_\Omega : \theta \mapsto J(\Omega(\theta))$ is differentiable, we define the shape derivative

$$< J'(\Omega), \theta > = < F'_\Omega, \theta >.$$

By the Hadamard structure theorem, it is known that the shape derivative is carried only on the boundary of the shape, i.e.,

$$< J'(\Omega), \theta > = \int_{\partial \Omega} j(\Omega) \theta \cdot n\, ds. \quad \text{(1)}$$

### 2.3 The boundary variation algorithm

To apply the gradient method to shape optimization, it remains to associate to a given shape derivative $J'(\Omega)$ a direction of slope $d$. To this end, it suffices to endow the space of vector fields from $\Omega$ into $\mathbb{R}^N$ with an Hilbert structure, for instance $H^1(\Omega)^N$. In this case, the descent direction is the unique element $d \in H^1(\Omega)^N$ such that for every $\theta \in H^1(\Omega)^N$,

$$\int_{\Omega} (\nabla d \cdot \nabla \theta + d \cdot \theta)\, dx = < J'(\Omega), \theta >.$$

Computing $d$ as the solution of (2) can also be interpreted as a regularization of the gradient.

**Remark 1** The choice of $H^1(\Omega)^N$ as space of variations is more dictated by technical considerations [it is easy to solve (2) with FreeFem+] rather than theoretical ones. Many choices could be made and it is not obvious to find the one that provides the better rate of convergence. Moreover, it is possible to use a more general framework and to define the direction $d$ as the minimizer (in some Banach space) of a functional of the form $\frac{1}{2} I(d) - < J'(\Omega), \theta >$, where $I$ is a positive functional. The present case corresponds to the choice $I(d) = \|d\|^2_{H^1(\Omega)}$.

**Remark 2** The Hadamard structure theorem tells us that the directional derivative of $J$ depends only on the value of the normal component $\theta \cdot n$ on $\partial \Omega$ [see formula (1)]. Thus, one can replace the space $H^1(\Omega)^N$ in (2) by $H^1(\partial \Omega)^N$ with the corresponding change of scalar product and the new descent direction is the solution of

$$\int_{\partial \Omega} d' \cdot \theta' + d \cdot \theta\, ds = < J'(\Omega), \theta >, \text{ for every } \theta \in H^1(\partial \Omega)^N,$$

where $'$ denotes the surface gradient. Nevertheless, it is more convenient to use (2) because it is simpler to solve, and it yields a natural extension of the mesh deformation on the whole domain $\Omega$.

The resulting algorithm can be summarized as follows:

1. Choose an initial shape $\Omega_0$.
2. Iteration until convergence for $n \geq 0$.
   a) Compute $d_n$ solution of the problem (2) with $\Omega = \Omega_n$.
   b) Set $\Omega_{n+1} = (\text{Id} - h_n d_n)(\Omega_n)$, where $h_n$ is a small, positive real.

### 2.4 Algorithmic details of the method

In the following, several algorithmic details are discussed, which make the method truly efficient and effective.

#### 2.4.1 Regularization of the mesh

It is well-known that numerical shape optimization may yield optimal designs with oscillating boundaries (i.e., having peaks and wells of length scale of the order of the mesh size). To avoid this problem, one can add a perimeter penalization to the cost function $J$, but the resulting solution will depend on the weight of the penalization. We prefer to use a regularization procedure that explicitly smooths the shape at each iteration. To this end, we introduce two different meshes of $\Omega$. At each iteration, a fine mesh $S_h$ is used to perform the finite element analysis and compute the descent direction $d$. We extract a coarser mesh $T_h$ from $S_h$, the nodes of which are moved in the direction $-d$ defined by (2). Finally, a new fine mesh $S_h$ is deduced from the displaced $T_h$ by mesh adaptation.

#### 2.4.2 Regularization of the shape gradient

The displacement usually presents singularities at the corners of the shape or at the changes of boundary conditions type. In such cases, the formula (5) is not correct anymore, as the