Steepest descent and Newton methods

using duality in

Structural Optimization

Håkan Johansson

December 20, 2002

Abstract

In Structural Optimization one typically seeks a minima of a given objective functional $\mathcal{I}$ which depends not only on the design parameters but also on the solution of a related equation of state of the form of a PDE. This makes the derivative of $\mathcal{I}$ difficult to calculate. In this thesis we study an adjoint methodology for solving optimization problems of this kind.

The main idea of this adjoint methodology is to introduce a Lagrangian functional $\mathcal{L}$ such that $\mathcal{L}$ has a stationary point where the objective functional $\mathcal{I}$ has a minima.

We can use this $\mathcal{L}$ for two different methods for finding a minima for $\mathcal{I}$. First we may use a steepest descent method where we use $\mathcal{L}$ to calculate the derivative of $\mathcal{I}$ in an efficient way. Secondly we may use Newton's method for finding stationary points for $\mathcal{L}$ directly.

We also study two test problems, one 1D and one 2D. For the 1D test problem the methodology works fine, but for the 2D test problem we encounter oscillating behavior in the design. We try some possible explanations and conclude that the most likely one is that the minima for the optimization problem is not well-defined.

keywords: structural optimization, FEM, optimal shape, shape derivative, steepest descent, Newton's method.

Preface

The work in this master thesis has been carried out at Chalmers Finite Element Center (Phi) at Chalmers University of Technology during the summer and fall of 2002.

I would like to express my great gratitude for the help and guidance from my examiner professor Kenneth Eriksson. His enthusiasm for learning new things is a great inspiration for me, and has been so from my first year at Chalmers a onward.

Parallel to my work, Fredrik Bengzon and Marcus Edstorp have been working on their theses. We have had many interesting discussions regarding optimization, FEM, elementary mathematics and life in general.

Dr Mats G Larson has also been involved in this work, since he advised Fredrik Bengzon whos work in bone remodeling has a very close connection to the work in this thesis.

Short discussions with interesting points from prof. Claes Johnson, prof. Michael Patriksson and prof. Kenneth Runesson also contributed to this work.

Finally I would like to thank all employees at Phi for help, encouragement and pleasant conversations.

Göteborg, December 2002

Håkan Johansson

Contents

1 Introduction

1.1 Objective

The purpose of this work is to study the performance of some natural computer based optimization methods for some basic engineering optimal design problems. The starting point for the work has been a draft article by Eriksson and Larson [6] in which duality arguments play a central role and are used both for the computation of the directions of steepest ascent/descent and for error analysis.

1.2 Background

In traditional optimal design, the engineer had to determine the optimization steps by pure intuition or by fairly crude estimates based on calculations by hand on simplified models of the situation at hand, and the resulting quality of the design had to be evaluated by "real life" tests. For example, in the automobile industry one had to build large numbers of cars just to destroy them in full scale crash tests.

With the introduction of modern computers much of the test phase can now be replaced by computer based simulations and recently Volvo/Ford released a new car model where essentially all crash tests in the development phase were done in computer simulations.

The next step is now to seek to involve the computer also in the very design process, and not only as a CAD tool or a test bench, but also to have the computer find out the path to an improved design, that is, to have the computer find the optimal design in an iterative process of computing the directions to go in each design step, determine the step size and then evaluate the result, and repeat to finally stop when the design criteria has been reached.

1.3 Structural Optimization

The aim of Structural Optimization is to find a design of a mechanical structure which is optimal from a certain (given) point of view. The most common may be to seek to minimize the deformation of the structure for a given load i.e. maximize the stiffness. Other objectives can be to minimize stresses or to control certain dynamical characteristics such as eigenmodes and eigenfrequencies.

Structural optimization can be divided into three fields, size-, shape- and topology optimization. In size optimization the control p describes a parameter in the governing state equation. The 1D test problem below is an example of this, where the control parameter corresponds to the thickness of a one-dimensional structure. In shape optimization the parameter describes the geometry of the domain. The 2D test problem below, where we seek the optimal design of a console is an example of this. In topology optimization the control, describes the existence/non-existence of material, which may result in holes or truss structures. We do not study topology optimization further, but we point out some connections between size-, shape- and topology optimization where suitable.

Throughout this thesis the state problems solved are second order partial differential equations (PDE). The Finite Element Method (FEM) is used for solving these.

1.4 Method

Two test problems have been formulated and studied. The solutions to these problems are intuitively known by forehand which gives some idea whether the solutions found are reasonable or not. During the work with these test problems the underlying methodology has become more clear and understandable.

1.5 Delimitations

The mathematical theory is not studied in detail because it requires a much deeper understanding in the theory for PDE. Studies of more sophisticated optimization algorithms are not done here either. No a priori or a posteriori error estimations done. In order to the text easier to read, most derivations are done in a discrete setting.

1.6 Disposition

This thesis is organized as follows, first we present a general methodology in an abstract framework in section 2, then in section 3 we apply the methodology to a 1D test problem, followed by a 2D test problem in section 4 where we encounter some problems which we try to explain. Then in section 5 follows some remarks on a matlab implementation and finally we have a general discussion and other possible applications in section 6 and future work in section 7.

2 Abstract formulation

The first three subsections follow in general Eriksson and Larson [6].

2.1 Problem formulation

The abstract form of the optimization problem reads: Find a control $p \in \mathcal{O}$ such that

$$\mathcal{I}(p, u(p)) = \min_{q \in \mathcal{O}} \mathcal{I}(q, u(q)) \tag{1}$$

where $\mathcal{I}$ is a given objective functional, $\mathcal{O}$ is the set of admissible controls and $u = u(p) \in V$ is the solution of the state equation

$$a(p; u, v) = l(p; v) \qquad \forall v \in V. \tag{2}$$

Here and in applications below the state equation is given in weak variational form with V a suitable linear test space and $l(p; \cdot)$ and $a(p; \cdot, \cdot)$ bounded linear and bilinear forms on V for $p \in \mathcal{O}$ with $a(p; \cdot, \cdot)$ symmetric and positive definite so that the state u is uniquely determined. For a solution p to the minimization problem (1) in the interior of $\mathcal{O}$ a necessary (first order) condition is

$$d_p \mathcal{I}(p, u(p)) = 0 \tag{3}$$

where d_p denotes a differentiation with respect to p.

More generally, for a local minima, not necessarily in the interior of $\mathcal{O}$, the necessary (first order) condition reads

$$\langle d_p \mathcal{I}(p, u(p)), q - p \rangle \geq 0 \qquad \forall q \in \mathcal{O}.$$

Here $\langle w, q \rangle$ denotes the scalar product-like paring of $q \in \mathcal{O} \subset \mathcal{Q}$ and w is the dual of $\mathcal{Q}$, where $\mathcal{Q}$ is a suitable linear space of controls. For p in the interior of $\mathcal{O}$ and $\delta p = q - p$ may represent all directions in $\mathcal{Q}$ so that in fact

$$\langle d_p \mathcal{I}(p, u(p)), \delta p \rangle = 0$$

for all $\delta p \in \mathcal{Q}$, which is the same as (3).

2.2 The continuous problem

Direct differentiation of $\mathcal{I}(p, u(p))$ gives

$$\langle d_p \mathcal{I}(p, u(p)), \delta p \rangle = \langle \partial_p \mathcal{I}(p, u), \delta p \rangle + \langle \partial_u \mathcal{I}(p, u), \delta u \rangle$$

where $u = u(p)$ and in the latter term $\delta u = d_p u \, \delta p$ and $\langle \cdot, \cdot \rangle$ denotes the duality paring of V and its dual. Thus for the total rate of change of $\mathcal{I}(p, u(p))$ with respect to p, we need to find, in particular, the rate of change $d_p u$

of u with respect to p. In principal, one way to do this is through direct differentiation of the state equation (2) with respect to p which gives

$$\partial_p a(p; u(p), v) + a(p; d_p u, v) = \partial_p l(p; v) \quad \forall v \in V. \tag{4}$$

However, to find $d_p u$ this way, involves first finding $u = u(p)$ by solving the state equation (2) and then solving (4) for $d_p u$ by solving a corresponding problem for each (infinitesimal) change of p. If p is discrete, for instance a vector, this is almost the same thing as numerically calculating $(\mathcal{I}(p+he_i, u(p+he_i)) - \mathcal{I}(p, u(p)))/h$, where $e_i = (0, 0, \dots, 1, 0, \dots, 0)$, measuring the rate of change with respect to for each component of this vector individually. Since this (in computing time) is a very expensive way of calculating the derivative of $\mathcal{I}(p, u(p))$ we consider an alternative procedure as follows. We first introduce the Lagrangian $\mathcal{L}$

$$\mathcal{L}(p, u, \lambda) = \mathcal{I}(p, u) + a(p; u, \lambda) - l(p; \lambda)$$

now considering p, u, λ as independent variables. Taking the derivative of $\mathcal{L}$ in the direction $\{\delta p,\ \delta u,\ \delta\lambda\}$ gives

$$\langle d\mathcal{L}, \{\delta p,\ \delta u,\ \delta\lambda\}\rangle = \langle \partial_p \mathcal{L}, \delta p\rangle + \langle \partial_u \mathcal{L}, \delta u\rangle + \langle \partial_\lambda \mathcal{L}, \delta\lambda\rangle$$

At a stationary point $\{p,\ u,\ \lambda\}$ of $\mathcal{L}$, we have that $\langle d\mathcal{L}, \{\delta p,\ \delta u,\ \delta\lambda\}\rangle$ is zero for all possible $\{\delta p,\ \delta u,\ \delta\lambda\}$. In particular this requires that

$$\langle \partial_\lambda \mathcal{L}, \delta\lambda\rangle = a(p; u, \delta\lambda) - l(p; \delta\lambda) = 0 \quad \forall \delta\lambda \in V$$

here it was used that $\langle \partial_\lambda a(p; u, \lambda), \delta\lambda\rangle = a(p; u, \delta\lambda)$ and $\langle \partial_\lambda l(p; \lambda), \delta\lambda\rangle = l(p; \delta\lambda)$. So $\mathcal{L}$ is stationary in λ if and only if u solves the state equation. Similarly the Lagrangian is stationary in u if and only if λ solves the adjoint equation

$$\langle \partial_u \mathcal{L}, \delta u\rangle = a(p; \delta u, \lambda) + \langle \partial_u \mathcal{I}, \delta u\rangle = 0 \quad \forall \delta u \in V \tag{5}$$

Finally, it follows from the just established facts, $a(p; u, \delta\lambda) - l(p; \delta\lambda) = 0$ and $a(p; \delta u, \lambda) + \langle \partial_u \mathcal{I}, \delta u\rangle = 0$, that

$$\begin{aligned}
\langle d_p\mathcal{I}(p,u(p)),\delta p\rangle &= \langle d_p\mathcal{I}(p,u),\delta p\rangle + \langle d_p \overbrace{(a(p;u,\lambda)-l(p;\lambda))}^{=0},\delta p\rangle \\
&= \langle \partial_p\mathcal{I}(p,u),\delta p\rangle + \langle \partial_u\mathcal{I}(p,u),\delta u\rangle + \langle \partial_p a(p;u,\lambda),\delta p\rangle \\
&\quad + \langle \partial_u a(p;u,\lambda),\delta u\rangle + \langle \partial_\lambda a(p;u,\lambda),\delta\lambda\rangle \\
&\quad - \langle \partial_p l(p;\lambda),\delta p\rangle - \langle \partial_\lambda l(p;\lambda),\delta\lambda\rangle \\
\{\langle\cdot,\cdot\rangle \text{ is linear}\} &= \overbrace{\langle \partial_p\mathcal{I}(p,u)+\partial_p a(p;u,\lambda)-\partial_p l(p;\lambda),\delta p\rangle}^{=\langle\partial_p\mathcal{L},\delta p\rangle} \\
&\quad + \underbrace{\langle \partial_u a(p;u,\lambda)+\partial_u\mathcal{I}(p,u),\delta u\rangle}_{=\langle\partial_u\mathcal{L},\delta u\rangle=0} \\
&\quad + \underbrace{\langle \partial_\lambda a(p;u,\lambda)-\partial_\lambda l(p;\lambda),\delta\lambda\rangle}_{=\langle\partial_\lambda\mathcal{L},\delta\lambda\rangle=0} \\
&= \langle d\mathcal{L},\{\delta p\ \delta u\ \delta\lambda\}\rangle
\end{aligned}$$

which shows two things, first that a stationary point in $\mathcal{I}$ with respect to p is a stationary point in $\mathcal{L}$ with respect to $\{p,\ u,\ \lambda\}$ if and only if the state and dual equations (2) and (5) are satisfied. The second thing shown is that

$$\langle d_p\mathcal{I}(p,u(p)),\delta p\rangle = \langle \partial_p\mathcal{I}(p,u)+\partial_p a(p;u,\lambda)-\partial_p l(p;\lambda),\delta p\rangle$$

and since this holds for all $\delta p \in \mathcal{O}$ the derivative of the objective functional can be computed using the identity

$$d_p\mathcal{I}(p,u(p)) = \partial_p\mathcal{I}(p,u)+\partial_p a(p;u,\lambda)-\partial_p l(p;\lambda)$$

where $u = u(p)$ and $\lambda = \lambda(p)$ are determined by (2) and (5) respectively.

2.3 Discrete formulation

Introducing the discretizations for the state, dual and control $u_h = \sum_i U_i\varphi_i$, $\lambda_h = \sum_i \Lambda_i\varphi_i$ and $p_h = \sum_i P_i\psi_i$, where $\{\varphi_i\}$ is a basis for the discrete state space $V^h \subset V$, with state and dual space for simplicity chosen to be the same and $\{\psi_i\}$ is a basis for the discrete control space $\mathcal{O}^h$. We may consider discretizing the state equation and the dual equation as follows

$$a(p_h;u_h,v) = l(p_h;v) \qquad \forall v\in V^h$$

$$a(p_h;\lambda_h,v) + \langle\partial_u\mathcal{I}(p_h,u_h),v\rangle = 0 \qquad \forall v\in V^h.$$

The derivative of the objective functional $\mathcal{I}(p_h,u_h(p_h))$ becomes

$$d_{P_i}\mathcal{I}(p_h,u_h) = \partial_{P_i}\mathcal{I}(p_h,u_h)+\partial_{P_i}a(p_h;u_h,\lambda_h)-\partial_{P_i}l(p_h;\lambda_h). \qquad (6)$$

2.4 Steepest descent

To minimize the objective functional $\mathcal{I}(p, u(p))$ the steepest descent algorithm may be used. Given a point p_h^k in $\mathcal{O}^h$, a better point p_h^{k+1} is found by taking a step in direction of the negative derivative of $\mathcal{I}(p_h, u_h(p_h))$. The derivative of the objective functional is calculated using the formula (6) above. Since the state u_h and the dual λ_h has to be calculated first, the resulting algorithm reads:

1. solve the state equation $a(p_h; u_h, v) = l(p_h; v)\ \forall v \in V^h$ for u_h.
2. solve the dual equation $a(p_h; \lambda_h, v) + \langle \partial_u \mathcal{I}, v\rangle = 0\ \forall v \in V^h$ for λ_h.
3. determine the search direction of steepest descent $-d_{P_i}\mathcal{I}(p_h, u_h)$ of $\mathcal{I}$ by calculating $d_{P_i}\mathcal{I}(p_h, u_h) = \partial_{p_i}\mathcal{I}(p_h, u_h) + \partial_{p_i}a(p_h; u_h, \lambda_h) - \partial_{p_i}l(p_h; \lambda_h)$
4. determine a step length α using a suitable line search algorithm.
5. update $p_h^{k+1} = p_h^k - \alpha d_{P_i}\mathcal{I}(p_h, u_h)$
6. repeat until convergence.

2.5 Newton's method

Since a stationary point p_h for the objective functional $\mathcal{I}(p_h, u_h(p_h))$, by introducing a suitable dual state λ_h as above, corresponds to a stationary Lagrangian

$$\langle d\mathcal{L}, \{\delta p,\ \delta u,\ \delta\lambda\}\rangle = 0$$

for all $\{\delta p,\ \delta u,\ \delta\lambda\}$, we may alternatively seek a minimizer p_h by seeking $\{p_h\ u_h\ \lambda_h\}$ such that

$$d\mathcal{L}(p_h, u_h, \lambda_h) = 0.$$

To find p_h (together with the corresponding u_h and λ_h) this way, it is natural to apply a Newton-type method by iterating with the Newton-step s^{k+1} (k being the iteration index) given by

$$d^2\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)s^{k+1} = -d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k).$$

Since here the residual $d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$ in general is nonzero during the iterations, the state and dual equations are not satisfied (i.e. an infeasible solution) before the method has converged. We may expand the residual by writing it in component form as

$$d\mathcal{L}(p_h, u_h, \lambda_h) = \begin{bmatrix} \partial_{P_i}\mathcal{L} \\ \partial_{U_i}\mathcal{L} \\ \partial_{\Lambda_i}\mathcal{L} \end{bmatrix} = \begin{bmatrix} \partial_{P_i}\mathcal{I} + \partial_{P_i}a(p_h; u_h, \lambda_h) - \partial_{P_i}l(p_h; \lambda_h) \\ a(p_h; \varphi_i, \lambda_h) + \langle \partial_{U_i}\mathcal{I}, \varphi_i\rangle \\ a(p_h; u_h, \varphi_i) - l(p_h; \varphi_i) \end{bmatrix}.$$

The Hessian $d^2\mathcal{L}(p_h, u_h, \lambda_h)$ is the second derivative of the Lagrangian. As usual, second derivatives may be taken in any order, that is $\partial_{pu}\mathcal{L} = \partial_{up}\mathcal{L}$ is true for all combinations of the derivatives (see for example Laumen [10]). The second derivatives may be collected in a block-symmetric matrix.

$$d^2\mathcal{L}(p_h, u_h, \lambda_h) = \begin{bmatrix} \partial^2_{P_i,P_j}\mathcal{L} & \partial^2_{P_i,U_j}\mathcal{L} & \partial^2_{P_i,\Lambda_j}\mathcal{L} \\ \partial^2_{U_i,P_j}\mathcal{L} & \partial^2_{U_i,U_j}\mathcal{L} & \partial^2_{U_i,\Lambda_j}\mathcal{L} \\ \partial^2_{\Lambda_i,P_j}\mathcal{L} & \partial^2_{\Lambda_i,U_j}\mathcal{L} & \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} \end{bmatrix}$$

Here, by our assumption that $a(p; u, v)$ is linear in u and v, so that in particular

$$\partial_{U_i} a(p_h; u_h, v) = \partial_{U_i} a(p_h; \sum_j U_j \varphi_j, v) = \partial_{U_i} \sum_j U_j a(p_h; \varphi_j, v) = a(p_h; \varphi_i, v),$$

and by the symmetry of $a(p_h; \cdot, \cdot)$ we have that $\partial^2_{U_i,\Lambda_j}\mathcal{L} = a(p_h; \varphi_i, \varphi_j) = \partial^2_{\Lambda_i,U_j}\mathcal{L}$ while $\partial^2_{P_i,U_j}\mathcal{L} = \partial^2_{P_i,U_j}\mathcal{I} + \partial_{P_i} a(p_h; \varphi_j, \lambda_h) = \partial^2_{U_j,P_i}\mathcal{L}$ also $\partial^2_{P_i,\Lambda_j}\mathcal{L} = \partial_{P_i} a(p_h; u_h, \varphi_j) - \partial_{P_i} l(p_h; \varphi_j) = \partial^2_{\Lambda_j,P_i}\mathcal{L}$ and $\partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} = 0$.

The algorithm when using a Newton-type method is (with current state, $r^k = \{p_h^k,\ u_h^k,\ \lambda_h^k\}^T$ and Newton-step $s^k = \{\delta p_h^k,\ \delta u_h^k,\ \delta\lambda_h^k\}^T$)

1. Choose a starting state r^0. One such choice may be r^0 such that it solves the state and the dual equations (has primal and dual feasibility).

2. Determine the residual $d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$ and the Hessian $d^2\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$

3. Calculate the Newton-step by solving
   $d^2\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)s^{k+1} = -d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$

4. Update the current state $r^{k+1} = r^k + s^{k+1}$

5. Repeat from (2) until convergence.

3 1D test problem: Hanging bar

As a first concrete example of an optimization problem of the above form we consider a hanging bar (see figure 1) that is rigidly attached at the upper end $x = 0$ and a force F is applied at the lower (free) end $x = L$. The bar has a varying cross-section with the area $p(x)$, and a density $\hat{\rho}$ which due to the gravity g gives a distributed load $p\hat{\rho}g$ along the x-axis. The objective is to minimize the displacement $u(L)$ at the free end, for a given total volume A_{tot} of the bar.

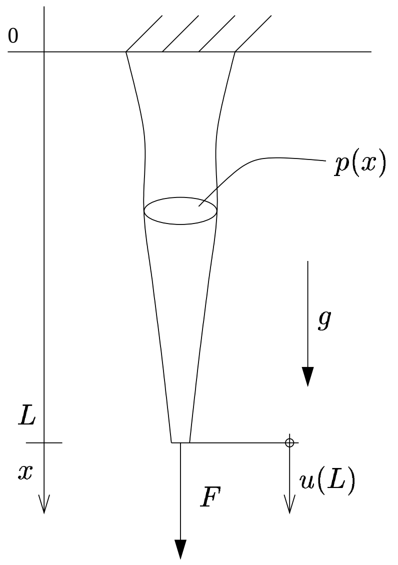

Figure 1: The force F applied at $x = L$

The mathematical formulation of the problem reads

$$\min_{p} \quad u(L)$$

subject to

$$\frac{d}{dx}\left(Ep\frac{du}{dx}\right) + p\rho = 0$$

$$u(0) = 0, \quad Ep\frac{du}{dx}\bigg|_{x=L} = F$$

$$\int_0^L p(x)\,dx = A_{tot} \quad p(x) > 0$$

where ρ is the specific density ($\rho = \hat{\rho}g$, $\hat{\rho}$ is the usual density and g is the gravity constant). The weak form of the state equation reads $a(p; u, v) = l(p; v) \;\; \forall v \in V$ with the bilinear form

$$a(p; u, v) = \int_0^L Ep\frac{du}{dx}\frac{dv}{dx}\,dx$$

and the linear form

$$l(p; v) = \int_0^L \rho p v \, dx + F v(L)$$

To enforce the side conditions on p a penalty approach may be used. This means adding a penalty functional $S(p)$ to the objective functional. If no penalization is used, the penalty functional is of course zero. The objective functional used below thus reads

$$\mathcal{I}(p, u(p)) = u(L) + S(p)$$

In order to calculate the total derivative of $\mathcal{I}(p, u(p))$ with respect to p, the dual methodology outlined above is used. The Lagrangian

$$\mathcal{L}(p, u, \lambda) = \mathcal{I}(p, u) + a(p; u, \lambda) - l(p; \lambda)$$

with the definitions above reads

$$\mathcal{L}(p, u, \lambda) = u(L) + S(p) + \int_0^L Ep \frac{du}{dx}\frac{d\lambda}{dx} \, dx - \int_0^L \rho p \lambda \, dx - F\lambda(L)$$

Setting the λ-derivative of the Lagrangian to zero gives the state equation $a(p; u, v) = l(p; v) \quad \forall v \in V$, here

$$\int_0^L Ep \frac{du}{dx}\frac{dv}{dx} \, dx = \int_0^L \rho p v \, dx + F v(L) \quad \forall v \in V, \tag{7}$$

and setting the u-derivative zero gives the dual equation
$a(p; v, \lambda) = -\langle \partial_u \mathcal{I}, v \rangle \quad \forall v \in V$, here with $\mathcal{I}(p, u(p)) = u(L) + S(p)$,

$$\int_0^L Ep \frac{dv}{dx}\frac{d\lambda}{dx} \, dx = -\int_0^L v\delta(L) \, dx = -v(L) \quad \forall v \in V, \tag{8}$$

where δ is the Dirac delta function The desired total p-derivative of the objective functional was

$$d_p \mathcal{I}(p, u(p)) = \partial_p \mathcal{I}(p, u) + \partial_p a(p; u, \lambda) - \partial_p l(p; \lambda). \tag{9}$$

Using discrete solutions of the state and dual equations $u_h = \sum_i U_i \varphi_i$, $\lambda_h = \sum_i \Lambda_i \varphi_i$ and the discretized control $p_h = \sum_i P_i \psi_i$ gives the terms in the discrete version of (9)

$$\partial_{P_i} \mathcal{I}(p_h, u_h) = \partial_{P_i} S(p_h),$$

$$\partial_{P_i} a(p_h; u_h, \lambda_h) = \int_0^L E\psi_i \frac{du_h}{dx}\frac{d\lambda_h}{dx} \, dx,$$

$$\partial_{P_i} l(p_h; \lambda_h) = \int_0^L \rho \psi_i \lambda_h \, dx,$$

and the total derivative of the objective functional reads

$$d_{P_i}\mathcal{I} = \partial_{P_i} S(p_h) + \int_0^L E\psi_i \frac{du_h}{dx}\frac{d\lambda_h}{dx}\,dx - \int_0^L \rho\psi_i\lambda_h\,dx.$$

Applying the steepest descent method to this problem we thus have to, in each step, first solve the discrete counterpart of (7) and (8) for u_h and λ_h, then compute the gradient $d_{P_i}\mathcal{I}$ and minimize in the direction of $-d_{P_i}\mathcal{I}$ using a line search. Numerical results using this procedure are presented below.

3.1 Newton's method

In the Lagrangian p, u, λ are independent. The penalty functional $S(p)$ depends only on p. The discrete Lagrangian reads

$$\mathcal{L}(p_h, u_h, \lambda_h) = u_h(L) + S(p_h) + \int_0^L Ep_h \frac{du_h}{dx}\frac{d\lambda_h}{dx}\,dx - \int_0^L \rho p_h\lambda_h\,dx - F\lambda_h(L)$$

The Newton residual was

$$d\mathcal{L}(p_h, u_h, \lambda_h) = \begin{bmatrix} \partial_{P_i}\mathcal{L} \\ \partial_{U_i}\mathcal{L} \\ \partial_{\Lambda_i}\mathcal{L} \end{bmatrix}.$$

Taking first derivatives of the Lagrangian we get the components of the Newton residual

$$\partial_{P_i}\mathcal{L} = \partial_{P_i} S(p_h) + \int_0^L E\psi_i \frac{du_h}{dx}\frac{d\lambda_h}{dx}\,dx - \int_0^L \rho\psi_i\lambda_h\,dx,$$

$$\partial_{U_i}\mathcal{L} = \int_0^L Ep_h \frac{d\varphi_i}{dx}\frac{d\lambda_h}{dx}\,dx + \varphi_i(L),$$

$$\partial_{\Lambda_i}\mathcal{L} = \int_0^L Ep_h \frac{du_h}{dx}\frac{d\varphi_i}{dx}\,dx - \int_0^L \rho p_h\varphi_i\,dx - F\varphi_i(L).$$

The Hessian in matrix format was

$$d^2\mathcal{L}(p_h, u_h, \lambda_h) = \begin{bmatrix} \partial^2_{P_i,P_j}\mathcal{L} & \partial^2_{P_i,U_j}\mathcal{L} & \partial^2_{P_i,\Lambda_j}\mathcal{L} \\ \partial^2_{U_i,P_j}\mathcal{L} & \partial^2_{U_i,U_j}\mathcal{L} & \partial^2_{U_i,\Lambda_j}\mathcal{L} \\ \partial^2_{\Lambda_i,P_j}\mathcal{L} & \partial^2_{\Lambda_i,U_j}\mathcal{L} & \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} \end{bmatrix}$$

and the components for this test problem are (remember the symmetry)

$$\partial^2_{P_i,P_j}\mathcal{L} = \partial^2_{P_iP_j} S(p_h),$$

$$\partial^2_{P_i,U_j}\mathcal{L} = \int_0^L E\psi_i \frac{d\varphi_j}{dx}\frac{d\lambda_h}{dx}\,dx,$$

$$\partial^2_{P_i,\Lambda_j}\mathcal{L} = \int_0^L E\psi_i \frac{du_h}{dx}\frac{d\varphi_j}{dx}\,dx - \int_0^L \rho\psi_i\varphi_j\,dx,$$

$$\partial^2_{U_i,U_j}\mathcal{L} = \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} = 0,$$

$$\partial^2_{U_i,\Lambda_j}\mathcal{L} = \int_0^L Ep_h \frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\,dx.$$

The Newton-step $s^{k+1} = \{\delta p_h^{k+1},\ \delta u_h^{k+1},\ \delta\lambda_h^{k+1}\}^T$ is acquired by solving the linear system

$$d^2\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)s^{k+1} = -d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$$

3.2 Dealing with side conditions; penalty approach

There are two side conditions on $p(x)$ to consider

$$\int_0^L p(x)\,dx = A_{tot} \quad \text{and} \quad p(x) > 0$$

The penalty for deviating from the total volume condition may be chosen as

$$S_1(p) = \beta\left(\int_0^L p(x)\,dx - A_{tot}\right)^2$$

where β is a penalty parameter. With a discrete version $p_h(x) = \sum_i P_i\psi_i$ of p the derivatives are

$$\partial_{P_i}S_1(p_h) = 2\beta\int_0^L \psi_i\,dx\left(\int_0^L p_h(x)\,dx - A_{tot}\right),$$

$$\partial^2_{P_iP_j}S_1(p_h) = 2\beta\int_0^L \psi_i\,dx\int_0^L \psi_j\,dx.$$

In order to prevent $p(x)$ to get too close to zero, and eventually become negative and thus probably interrupt the optimization algorithm, one could try adding a penalty of the form

$$S_2(p) = \epsilon\int_0^L \frac{1}{p(x)}\,dx$$

where ϵ is another penalty parameter. After discretization the derivatives are

$$\partial_{P_i}S_2(p_h) = -\epsilon\int_0^L \frac{\psi_i}{(p_h(x))^2}\,dx,$$

$$\partial^2_{P_iP_j}S_2(p_h) = 2\epsilon\int_0^L \frac{\psi_i\psi_j}{(p_h(x))^3}\,dx.$$

This penalty term grows rapidly close to $p = 0$, so the step in the optimization algorithm must be small close to $p = 0$. In the test problem under consideration small values of p leads to large deformations, resulting in large displacements at the free end, so in general p will stay positive anyway. Therefore the constraint can be omitted in this case (i.e. $\epsilon = 0$).

When using more than one penalty functionals the total penalty functional is

$$S(p) = S_1(p) + S_2(p).$$

3.3 Dealing with side conditions; strongly imposed via a projected gradient

The side conditions for the total volume and positive values of the control may be enforced by using a projection-like technique. Below, when we refer to "projected gradient" this is what we mean: Let δP_i be the desired change in each nodal value P_i and add to this change and a constant a, which is the same for all control nodes (i). The value of a is chosen such that the total volume is A_{tot}, that is

$$\sum_i (P_i + \delta P_i + a) \int_0^L \psi_i \, dx = A_{tot}.$$

Solving for a gives

$$a = \frac{A_{tot} - \sum_i (P_i + \delta P_i) \int_0^L \psi_i \, dx}{\sum_i \int_0^L \psi_i \, dx}.$$

The new nodal values of p_h are given by $P_i^{k+1} = P_i^k + \delta P_i + a$. If this results in negative values for some P_i^{k+1}, these are set to a minimal allowed value and a new a is calculated, but this time only for the remaining P_i^{k+1}. By enforcing the side conditions this way, the penalty term $S(p_h)$ may be omitted. This technique can be viewed as a penalty approach with infinitely large penalty parameters without the numerical problems involved with such an approach.

3.4 Line search

In the steepest descent algorithm the step length α is determined by a line search. The line search used here is very simple. First a step length of one is tested. The objective functional is calculated, and if the value of the objective functional improves the step is accepted. If it is not, then the step length halved is successively divided by two until a descent step is found. If the step length has to be halved 50 times, it is assumed that line search has converged.

3.5 Results for the 1D test problem

For simplicity we have considered piecewise constant (discontinuous) control $p_h = \sum_i P_i \psi_i$ on a uniform partition of [0,L] along the one-dimensional bar under consideration with 15 elements, while the state and the dual are discretized with piecewise linear basis functions. Figure 2 below shows the converged optimal piecewise constant control (cross section area) p_h as a function of x along the bar with $L = 1$, $\rho = 1$, $F = 0.2$, $E = 1$ and $\beta = 10000$. Here we used Newton's method, but also the Steepest descent method perform as desired in this case. The optimal design p_h is concave which seems reasonable under the given conditions. Note that for $\rho = 0$ one expects the optimal distribution of volume to be with a uniform $p_h \equiv 1$ cross area, while with $F = 0$ the optimal solution degenerate by putting all the mass at $x = 0$.

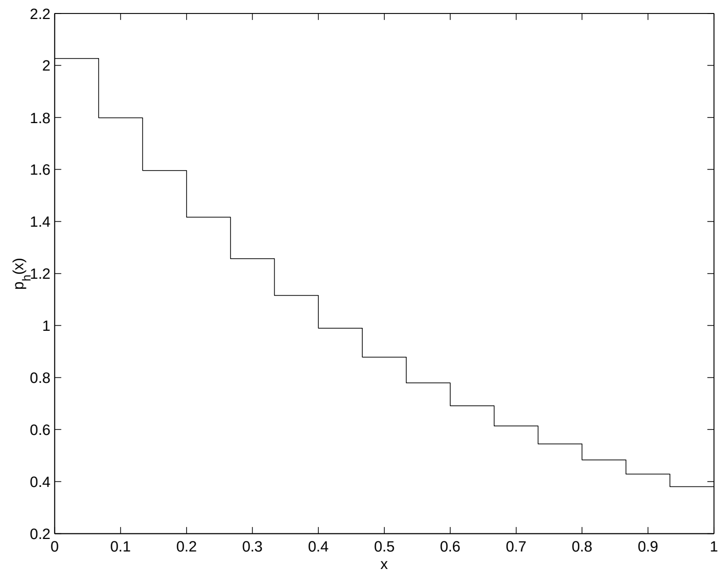

Figure 2: Optimal $p_h(x)$ (Newton's method)

3.5.1 Steepest descent

We start with a bar with uniform cross-section area, $P_i = 1$. In each iteration the computing time, number of line search steps (L S step), change in the displacement, change in the control P_i and the displacement at the end is calculated. The results are found in table 1. For comparison, machine$_\epsilon$ = $2.2204E - 16$. The method converges, but rather slowly because the changes in the control P_i and the end displacement $U(L)$ decrease slowly. These should be close to zero when finished. The slow convergence depends partly

Iter	Time	L S step	$\Delta U(L)$	$\|\|\Delta P_i\|\|$	$U(L)$
1	1.468	1	-1.308E-01	1.394E+00	0.5692
2	4.403	1	-7.283E-03	3.429E-01	0.5619
3	6.755	2	-1.235E-03	1.075E-01	0.5607
4	8.830	2	-6.369E-04	7.150E-02	0.5601
5	10.918	2	-3.594E-04	5.686E-02	0.5597
6	12.992	2	-2.212E-04	5.254E-02	0.5595
7	15.229	3	-1.964E-04	3.461E-02	0.5593
8	17.276	2	-1.315E-04	4.172E-02	0.5592
9	19.380	3	-1.014E-04	2.565E-02	0.5591
10	21.460	2	-6.227E-05	2.325E-02	0.5590
11	23.528	2	-4.322E-05	2.322E-02	0.5590
12	25.669	3	-3.984E-05	1.667E-02	0.5589
13	27.753	2	-2.767E-05	1.626E-02	0.5589
14	29.778	2	-1.807E-05	1.693E-02	0.5589
15	31.937	3	-2.179E-05	1.240E-02	0.5588
⋮	⋮	⋮	⋮	⋮	⋮
84	176.271	1	-1.366E-14	6.359E-07	0.5588
85	178.543	4	-1.410E-14	1.462E-07	0.5588
86	181.760	11	0.000E+00	5.021E-10	0.5588

Table 1: Results for Steepest descent, projected gradient

on the choice of line search method, because it finds a step length that decreases the objective functional instead of a step length that minimizes the objective functional.

3.5.2 Newton's method

Again we start with uniform thickness ($P_i = 1$). Since the state and the dual equations are not solved explicitly in each iteration, the state solution U^* corresponding to the current p is calculated for comparison, so that one can see how far the current state is from a feasible one. The other quantities measured are the same as in the steepest descent method. The results are collected in table 2. Note that the changes in control and displacement at the lower end decreases almost quadratically from iteration 2 to iteration 7. Quadratic convergence indicates that Newton's method works properly and that the minima is well-defined.

3.6 Conclusion 1D test problem

This is a quite straightforward problem with a quite well-defined minima. Both steepest descent and Newton's method works fine, but compared to steepest descent Newton's method is faster both in time and in number of

Iter	Time	$\Delta U(L)$	$\|\|\Delta P_i\|\|$	$\|\|\partial_{P_i}\mathcal{L}\|\|$	$\|\|U - U^*\|\|$	$U(L)$
1	1.052	-2.810E-01	2.087E+00	1.574E-01	1.305E+00	0.4190
2	4.163	1.062E-01	1.747E+00	4.904E-02	2.039E-01	0.5252
3	5.901	3.274E-02	6.061E-01	1.689E-02	1.058E-01	0.5579
4	7.484	8.282E-04	4.422E-02	7.203E-04	6.240E-03	0.5588
5	9.237	-4.366E-06	2.225E-04	1.522E-05	2.142E-05	0.5588
6	10.830	1.130E-09	2.113E-08	1.656E-09	1.381E-09	0.5588
7	12.464	-7.680E-16	6.638E-15	3.636E-13	2.073E-14	0.5588
8	14.221	-1.445E-15	5.263E-15	3.633E-13	1.251E-14	0.5588

Table 2: Results for Newton's method

iterations. This may change (but not likely) if more efficient algorithms for the steepest descent (e.g. like using a conjugated gradient) and Newton's method (e.g. reduced Hessian) is used. We can be quite confident that the adjoint methodology for the derivative of the objective functional is correct, and since Newton's method works fine, a stationary state to the Lagrangian is a minima to the objective functional. We conclude that nothing in this test problem contradict the methodology presented in the abstract setting.

4 2D test problem: The console

We now turn to a more interesting problem, seemingly not very complicated, but as it turns out, in fact quite challenging. We consider a console viewed as a 2D domain $\Omega(p)$ with the left side ($x = 0$) rigidly attached (homogeneous Dirichlet boundary) and the other sides $\Gamma_{x>0}$ are free (homogeneous Neumann boundary). On the upper right corner $(L, 0)$ a point load g_0 is applied. The gravity is assumed to have no influence. The control $p(x)$ describes the thickness in the y-direction with the upper boundary at $y = 0$ and the lower boundary Γ_p determined by the thickness p. The console is modeled with linear elasticity (Naviers equations) with plan strain condition. For a very short summary of linear elasticity see appendix A. Under the condition that the total area of the domain has a given value A_{tot}, find $p(x)$ so that the y-component u_y of the displacement $u = (u_x, u_y)$ at the right end $(L, 0)$ is minimized. The console is also described in figure 3. The problem has the mathematical formulation

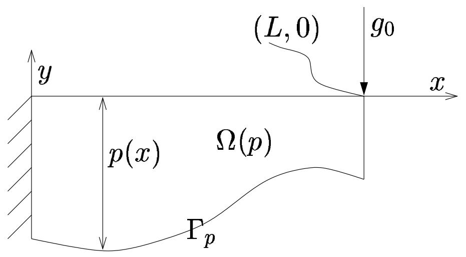

Figure 3: The force g_0 applied at $(L, 0)$

$$\min_p \; -u_y(L,0) \tag{10}$$

$$\int_{\Omega(p)} \sigma(u) : \epsilon(v)\, d\Omega = \int_{\Omega(p)} f \cdot v\, d\Omega + \int_{\Gamma_{x>0}} g \cdot v\, ds, \quad u(0,y) = 0$$

$$\int_0^L p(x)\, dx = A_{tot}, \quad p(x) > 0.$$

Again we have a state equation of the form $a(p; u, v) = l(p; v)$, now with the bilinear form

$$a(p; u, v) = \int_{\Omega(p)} \sigma(u) : \epsilon(v)\, d\Omega$$

and the linear form

$$l(p; v) = \int_{\Omega(p)} f \cdot v\, d\Omega + \int_{\Gamma_{x>0}} g \cdot v\, ds$$

Here $\epsilon(u)$ is the usual strain tensor and $\sigma(u)$ is the corresponding stress tensor, which are related via Young's modulus of elasticity E and Poisson's ratio ν in Hooke's law. Again, for a short summary, se appendix A. We have no volume load f, (due to no gravity), furthermore g is a point load of the form $g_0(0, -\delta(0, L))$, where δ is Dirac's delta function. Integrating this gives the resulting linear form

$$l(p; v) = -g_0 v_y(L, 0)$$

The objective functional under consideration is (with a penalty term $S(p)$ added for generality)

$$\mathcal{I}(p, u(p)) = -u_y(0, L) + S(p)$$

and the abstract Lagrangian is (as before p, u, λ are independent)

$$\mathcal{L}(p, u, \lambda) = \mathcal{I}(p, u) + a(p; u, \lambda) - l(p; \lambda)$$

with the abstract derivative

$$d_p \mathcal{I}(p, u(p)) = \partial_p \mathcal{I} + \partial_p a(p; u, \lambda) - \partial_p l(p; \lambda) \tag{11}$$

where u is obtained by solving the state equation

$$\int_{\Omega(p)} \sigma(u) : \epsilon(v)\, d\Omega = -g_0 v_y(L, 0) \quad \forall v \in V \tag{12}$$

and λ comes from the dual equation

$$\int_{\Omega(p)} \sigma(v) : \epsilon(\lambda)\, d\Omega = -\int_{\Omega(p)} (0, -\delta(L, 0)) \cdot v\, d\Omega = v_y(L, 0) \quad \forall v \in V \tag{13}$$

since $\partial_u \mathcal{I} = (0, -\delta(L, 0))$. We now continue our derivations in the discrete setting with $u_h = \sum_i U_i \varphi_i$, $\lambda_h = \sum_i \Lambda_i \varphi_i$ and $p_h = \sum_i P_i \psi_i$. The first and the last of the terms in the discrete counterpart of (11) is

$$\partial_{P_i} \mathcal{I}(p_h, u_h(p_h)) = \partial_{P_i} S(p_h)$$

$$\partial_{P_i} l(p_h; \lambda_h) = 0$$

since $l(p_h; \lambda_h)$ is in fact independent of p_h in this case. The p-derivative of the bilinear form is somewhat more complicated. In appendix B we show that

$$\partial_{P_i} a(p_h; u_h, \lambda_h) = \int_0^L [\sigma(u_h) : \epsilon(\lambda_h)]_{\Gamma_{p_h}} \psi_i\, dx \tag{14}$$

where $[\ \cdot\ ]_{\Gamma_{p_h}}$ means evaluation at $\Gamma_{p_h} = \{(x, -p_h(x)) : 0 < x < L\}$. The total derivative of the objective functional is then

$$\partial_{P_i} \mathcal{I}(p_h, u_h(p_h)) = \partial_{P_i} S(p_h) + \int_0^L [\sigma(u_h) : \epsilon(\lambda_h)]_{\Gamma_{p_h}} \psi_i\, dx$$

where u_h and λ_h is obtained by solving the discrete counterparts of (12) and (13).

4.1 Newton, penalty approach

In Newton's method, stationary points of the Lagrangian are sought. We find (p_h, u_h, λ_h) so that the derivative of the Lagrangian is zero by iterating with

$$d^2\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)s^{k+1} = -d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$$

where $s^k = \{\delta p^k,\ \delta u^k,\ \delta\lambda^k\}^T$. The Lagrangian for this test problem is

$$\mathcal{L} = -u_y(0, L) + S(p_h) + \int_{\Omega(p_h)} \sigma(u_h) : \epsilon(\lambda_h)\, d\Omega + g_0\lambda_{yh}(L)$$

The residual $d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k)$ for the Lagrangian becomes

$$d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k) = \begin{bmatrix} \partial_{P_i}\mathcal{L} \\ \partial_{U_i}\mathcal{L} \\ \partial_{\Lambda_i}\mathcal{L} \end{bmatrix}$$

with the terms

$$\partial_{P_i}\mathcal{L} = \int_0^L [\sigma(u_h) : \epsilon(\lambda_h)]_{\Gamma_{p_h}}\ \psi_i\, dx + \partial_{P_i}S(p_h)$$

$$\partial_{U_i}\mathcal{L} = \int_{\Omega(p_h)} \sigma(\varphi_i) : \epsilon(\lambda_h)\, d\Omega - \varphi_{yi}(L, 0)$$

$$\partial_{\Lambda_i}\mathcal{L} = \int_{\Omega(p_h)} \sigma(u_h) : \epsilon(\varphi_i)\, d\Omega + g_0\varphi_{yi}(L, 0).$$

The Hessian contains the second derivatives

$$d^2\mathcal{L}(p_h, u_h, \lambda_h) = \begin{bmatrix} \partial^2_{P_i,P_j}\mathcal{L} & \partial^2_{P_i,U_j}\mathcal{L} & \partial^2_{P_i,\Lambda_j}\mathcal{L} \\ \partial^2_{U_i,P_j}\mathcal{L} & \partial^2_{U_i,U_j}\mathcal{L} & \partial^2_{U_i,\Lambda_j}\mathcal{L} \\ \partial^2_{\Lambda_i,P_j}\mathcal{L} & \partial^2_{\Lambda_i,U_j}\mathcal{L} & \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} \end{bmatrix}$$

with the terms (note the symmetry)

$$\partial^2_{P_i,P_j}\mathcal{L} = \partial_{P_i,P_j}S(p_h) - \int_0^L [\partial_y\, (\sigma(h_h) : \epsilon(\lambda_h))]_{\Gamma_{p_h}}\ \psi_i\psi_j\, dx$$

$$\partial^2_{P_i,U_j}\mathcal{L} = \int_0^L [\sigma(\varphi_j) : \epsilon(\lambda_h)]_{\Gamma_{p_h}}\ \psi_i\, dx$$

$$\partial^2_{P_i,\Lambda_j}\mathcal{L} = \int_0^L [\sigma(u_h) : \epsilon(\varphi_j)]_{\Gamma_{p_h}}\ \psi_i\, dx$$

$$\partial^2_{U_i,P_j}\mathcal{L} = \int_0^L [\sigma(\varphi_i) : \epsilon(\lambda_h)]_{\Gamma_{p_h}}\ \psi_j\, dx$$

$$\partial^2_{U_i,U_j}\mathcal{L} = \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} = 0$$

$$\partial^2_{U_i,\Lambda_j}\mathcal{L} = \int_{\Omega(p)} \sigma(\varphi_i) : \epsilon(\varphi_j)\, d\Omega = \partial^2_{\Lambda_i,U_j}\mathcal{L}$$

$$\partial^2_{\Lambda_i,P_j}\mathcal{L} = \int_0^L [\sigma(u_h) : \epsilon(\varphi_i)]_{\Gamma_{p_h}}\, \psi_j\, dx.$$

The side conditions and the discretization look exactly the same as in the 1D test problem (although p describes something different here). The penalty terms introduced in subsection 3.2 can be used again.

4.2 Newton, Lagrangian multipliers

Instead of a penalty or projection approach we may also enforce the side conditions on p via Lagrangian multipliers. The area constraint is imposed via a Lagrangian multiplier η and the side condition $p(x) > 0$ is here dealt with introducing a new discretization for the control of the form

$$p_h(x) = \sum_i e^{P_i}\psi_i.$$

Corresponding to a change of variables $P_i \to e^{P_i}$, the thickness of the console at node i is now e^{P_i}. Because of this the new P_i can take any value without violating the side condition of positive p. The state equation can also be viewed as a constraint and it has already been included in the Lagrangian, so it is natural to continue and include the area constraint as well. The new Lagrangian is

$$\mathcal{L} = -u_y(0,L)+S(p_h)+\int_{\Omega(p_h)} \sigma(u_h) : \epsilon(\lambda_h)\, d\Omega+g_0\lambda_h(L)+\eta(\int_0^L \sum_i e^{P_i}\psi_i-A_{tot}).$$

A stationary point to the new Lagrangian is found by the usual Newton iteration.

$$d^2\mathcal{L}(p_h^k, u_h^k, \lambda_h^k, \eta^k)s^{k+1} = -d\mathcal{L}(p_h^k, u_h^k, \lambda_h^k, \eta^k)$$

where $s^k = \{\delta p^k,\ \delta u^k,\ \delta\lambda^k,\ \delta\eta^k\}^T$. The algorithm is the same as for Newton's method described in subsection 2.5. The residual $d\mathcal{L}(p_h, u_h, \lambda_h, \eta)$ is

$$d\mathcal{L}(p_h, u_h, \lambda_h, \eta) = \begin{bmatrix} \partial_{P_i}\mathcal{L} \\ \partial_{U_i}\mathcal{L} \\ \partial_{\Lambda_i}\mathcal{L} \\ \partial_{\eta}\mathcal{L} \end{bmatrix}$$

with the terms (for the derivation of the first term of $\partial_{P_i}\mathcal{L}$, see appendix B)

$$\partial_{P_i}\mathcal{L} = e^{P_i}\int_0^L [\sigma(u_h) : \epsilon(\lambda_h)]_{\Gamma_{p_h}}\, \psi_i\, dx + \eta(e^{P_i}\int_0^L \psi_i)$$

$$\partial_{U_i}\mathcal{L} = \int_{\Omega(p)} \sigma(\varphi_i) : \epsilon(\lambda_h)\, d\Omega - \varphi_{yi}(L,0)$$

$$\partial_{\Lambda_i}\mathcal{L} = \int_{\Omega(p)} \sigma(u_h) : \epsilon(\varphi_i)\, d\Omega + g_0 \varphi_{yi}(L,0)$$

$$\partial_\eta \mathcal{L} = \int_0^L \sum_i e^{P_i}\psi_i - A_{tot}$$

and the Hessian $d^2\mathcal{L}(p_h, u_h, \lambda_h, \eta)$ is

$$d^2\mathcal{L}(p_h, u_h, \lambda_h, \eta) = \begin{bmatrix} \partial^2_{P_i,P_j}\mathcal{L} & \partial^2_{P_i,U_j}\mathcal{L} & \partial^2_{P_i,\Lambda_j}\mathcal{L} & \partial^2_{P_i,\eta}\mathcal{L} \\ \partial^2_{U_i,P_j}\mathcal{L} & \partial^2_{U_i,U_j}\mathcal{L} & \partial^2_{U_i,\Lambda_j}\mathcal{L} & \partial^2_{U_i,\eta}\mathcal{L} \\ \partial^2_{\Lambda_i,P_j}\mathcal{L} & \partial^2_{\Lambda_i,U_j}\mathcal{L} & \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} & \partial^2_{\Lambda_i,\eta}\mathcal{L} \\ \partial^2_{\eta,P_j}\mathcal{L} & \partial^2_{\eta,U_j}\mathcal{L} & \partial^2_{\eta,\Lambda_j}\mathcal{L} & \partial^2_{\eta,\eta}\mathcal{L} \end{bmatrix}$$

with the terms (using the symmetry)

$$\partial^2_{P_i,P_j}\mathcal{L} = -e^{P_i}e^{P_j}\int_0^L [\partial_y(\sigma(u_h) : \epsilon(\lambda_h))]_{\Gamma_{p_h}} \psi_i\psi_j\, dx + \delta_{ij}\eta(e^{P_i}\int_0^L \psi_i\, dx)$$

where $\delta_{ij} = 1$ for $i = j$ and 0 for $i \neq j$ and the derivation of the first term can be found in appendix B

$$\partial^2_{P_i,U_j}\mathcal{L} = e^{P_i}\int_0^L [\sigma(\varphi_j) : \epsilon(\lambda_h)]_{\Gamma_{p_h}} \psi_i\, dx$$

$$\partial^2_{P_i,\Lambda_j}\mathcal{L} = e^{P_i}\int_0^L [\sigma(u_h) : \epsilon(\varphi_j)\psi_i]_{\Gamma_{p_h}}\, dx$$

$$\partial^2_{P_i,\eta}\mathcal{L} = e^{P_i}\int_0^L \psi_i\, dx$$

$$\partial^2_{U_i,\Lambda_j}\mathcal{L} = \int_{\Omega(P)} \sigma(\varphi_i) : \epsilon(\varphi_j)\, d\Omega$$

$$\partial^2_{U_i,U_j}\mathcal{L} = \partial^2_{U_i,\eta}\mathcal{L} = \partial^2_{\Lambda_i,\Lambda_j}\mathcal{L} = \partial^2_{\Lambda_i,\eta}\mathcal{L} = \partial^2_{\eta,\eta}\mathcal{L} = 0$$

The technique with Lagrangian multipliers for the constraints is in fact a special case of a more general method for solving minimization problems with side conditions. That method is called Sequential Quadratic Programming, SQP, and a short description of SQP in a general setting can be found in appendix C.

4.3 Implementation in matlab

For this test problem two matlab implementations have been made. One using bilinear elements and one using triangular elements. It seems that the quality of the mesh (i.e. how distorted the elements are) has an influence on the shape derivative. In this aspect a bilinear element with an isoparametric mapping, 2×2 Gauss quadrature seems to bring some advantages as compared to a triangular mesh. For the steepest descent algorithm we use the line search described in subsection 3.4, but the shape derivative is normalized first. The side conditions on p are enforced strongly as described in 3.3.

When the lower boundary Γ_{p_h} moves during the iterations the FE-mesh has to be updated. For bilinear elements this is done by simply updating the y-component for each node so that the position relative to the upper and lower boundary is unchanged, a sort of "squeezing" process. Since the mesh for bilinear elements is structured this works fine. For a triangular mesh this method works only for small changes in Γ_{p_h}, but for large movements on the lower boundary the elements gets distorted and can even turn into an upside-down position. To come around this a new mesh is generated in every iteration while the "squeezer method" is used in the line search. This may result in the line search suggesting a step-length that does not decrease the objective functional. Further, when a new mesh is generated, the state u_h and the dual λ_h has to be updated. This is done by solving the state and dual equations again. In the Newton method this results in that the current state is always feasible.

4.4 Steepest descent bilinear elements

We start with 16 control nodes and the steepest descent algorithm, with line search. Parameter values are Young's module of elasticity $E = 1000$, Poisson's ratio $\nu = 0.3$, the load at the end $g_0 = 1$, console length $L = 4$ and the total area is $A_{tot} = 4$. These values are the same for all tests below. For the bilinear elements the results can be found in table 3 and the resultin shape in figure 4 It can be seen that very little happens after iteration 4.

Iter	Time	L S step	$\Delta U(L,0)$	$\|\|\Delta P_i\|\|$	$U(L,0)$
0	0	-	-	-	0.2364
1	4.169	0	-6.707E-02	1.311E+00	0.1693
2	10.857	1	-1.530E-02	4.783E-01	0.1540
3	17.857	2	-1.059E-03	1.576E-01	0.1530
4	25.610	3	-4.624E-05	6.561E-02	0.1529
5	57.021	32	-5.854E-14	1.143E-10	0.1529
6	91.222	35	-4.891E-14	1.428E-11	0.1529
7	138.860	50	8.965E-14	4.264E-16	0.1529

Table 3: Results for Steepest descent, projected gradient

The general shape seems intuitively correct, but there is a strange oscillating behavior at the left end. Why this happens is not quite clear to us, but we present some possible explanations below.

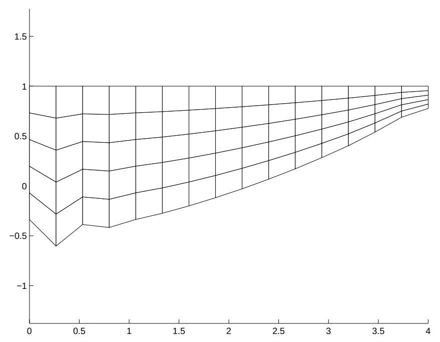

Figure 4: The shape after 8 iterations

4.5 Steepest descent triangular elements

Again with 16 control nodes, using the steepest descent algorithm, with line search we get the final shape figure 5 and the computed results in table 4. Here almost nothing happens after iteration 7.

Iter	Time	L S step	$\Delta U(L,0)$	$\|\|\Delta P_i\|\|$	$U(L,0)$
0	0	-	-	-	0.2380
1	11.627	0	-5.136E-02	6.610E-01	0.1866
2	22.652	0	-3.091E-03	5.295E-01	0.1848
3	32.234	0	-2.345E-02	4.052E-01	0.1629
4	43.008	2	-6.543E-04	1.829E-01	0.1633
5	54.265	3	-9.976E-06	4.146E-02	0.1633
6	61.086	0	-1.816E-03	1.874E-01	0.1619
7	77.443	5	-2.686E-05	2.278E-02	0.1644
8	139.850	33	-7.633E-15	7.926E-11	0.1644

Table 4: Results for Steepest descent, projected gradient

Here almost nothing happens after iteration 7 and the strange behavior at the left end occurs again. This phenomenon has also been seen when solving this test problem using matlab's built in function *fmincon*, which solves a general constrained optimization problem where the derivative of the objective function is calculated using numerical differentiation.

4.6 Modifying the gradient

We start our search for an explanation for the phenomenon at the left end by studying the derivative of the objective functional. When calculating the total derivative of the objective functional, the derivative of the bilinear form

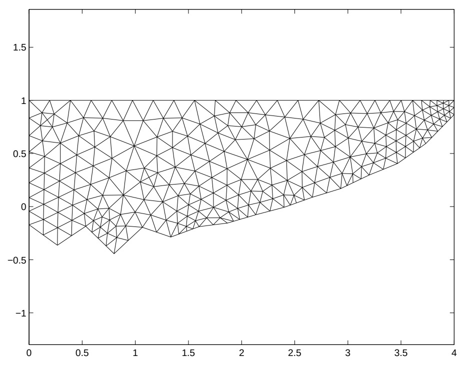
Figure 5: The shape after 10 iterations

gives the most important contribution.

$$\partial_{P_i} a(p_h; u_h, \lambda_h) = \int_0^L [\sigma(u_h) : \epsilon(\lambda_h)]_{\Gamma_{p_h}} \psi_i \, dx$$

When using linear basis functions ψ_i for the control, the first and the last of the basis functions are supported on just one interval, while the rest are supported on two adjacent intervals. Consequently the derivatives with respect to P_1 and P_N (N is the number of control nodes) are about half as large as the derivatives P_2, P_{N-1} respectively. This motivates the use of a modified gradient where the first and the last terms are multiplied by two, at least if the quantity $\sigma(u_h) : \epsilon(\lambda_h)$ is of the same size along Γ_p. For a uniform thickness of the console, the shape derivative is plotted in figure 6 with unmodified gradient on the left and a modified gradient on the right. We note that at a minima, the gradient is zero, so this modification should not change the optima, just the way to get there.

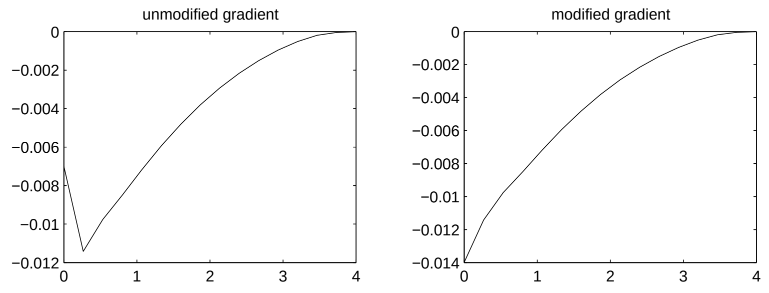

Figure 6: The derivative of the bilinear form when the console has uniform thickness.

We try this modified gradient for bilinear elements with the results in table 5 and final shape in figure 7

and for triangular elements the result and final shape can be found in table 6 and figure 8 respectively.

Iter	Time	L S step	$\Delta U(L,0)$	$\|\|\Delta P_i\|\|$	$U(L,0)$
0	0	-	-	-	0.2364
1	4.190	0	-6.605E-02	1.364E+00	0.1703
2	13.009	1	-1.584E-02	4.957E-01	0.1545
3	20.110	2	-1.354E-03	1.573E-01	0.1531
4	29.010	4	-3.687E-05	3.215E-02	0.1531
5	41.159	8	-9.366E-08	1.914E-03	0.1531
6	70.574	28	-3.314E-14	1.820E-09	0.1531
7	106.306	36	-1.155E-14	7.109E-12	0.1531
8	136.342	29	-2.143E-14	9.099E-10	0.1531

Table 5: Results for Steepest descent, projected gradient

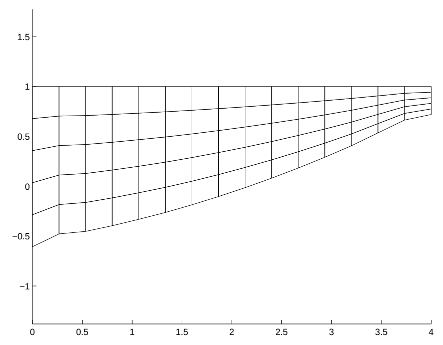

Figure 7: The shape after 8 iterations

We see that using the modified gradient gives a more smooth shape and a smaller displacement at the end, but a small disruption remain.

4.7 Regularization

It seems that the strange behavior at the left end remains. A possibility is that the minima is not well-defined, making it difficult to find the optimal p. Therefore a penalization for abrupt changes in the first derivative of the control p (large second derivatives) may be introduced. It can be viewed as a regularization. Introduce the regularization penalty $S_R(p_h)$ with the penalty parameter α. In the discrete setting with linear basis functions for the control where $\partial_x p_h$ and Δx is elementwise and P is nodewise so that Δx_i is bounded by P_i and P_{i+1}. The penalty term is chosen as

$$S_R(p_h) = \alpha \sum_{i=2}^{N-1} \left(\left(\frac{\partial p_h}{\partial x} \right)_{i+1} - \left(\frac{\partial p_h}{\partial x} \right)_i \right)^2 = \alpha \sum_{i=2}^{N-1} \left(\frac{P_{i+1} - P_i}{\Delta x_i} - \frac{P_i - P_{i-1}}{\Delta x_{i-1}} \right)^2$$

Iter	Time	L S step	$\Delta U(L,0)$	$\|\|\Delta P_i\|\|$	$U(L,0)$
0	0	-	-	-	0.2380
1	5.780	0	-6.019E-02	7.424E-01	0.1778
2	15.055	0	-2.005E-02	5.058E-01	0.1576
3	22.251	0	-5.845E-03	3.637E-01	0.1522
4	29.457	0	-1.502E-04	1.772E-01	0.1522
5	39.797	2	-1.389E-04	7.697E-02	0.1526
6	120.338	50	-4.247E-14	9.710E-16	0.1521
7	181.490	37	-1.011E-13	1.769E-12	0.1521
8	240.595	36	-5.262E-14	3.539E-12	0.1521

Table 6: Results for Steepest descent, projected and modified gradient

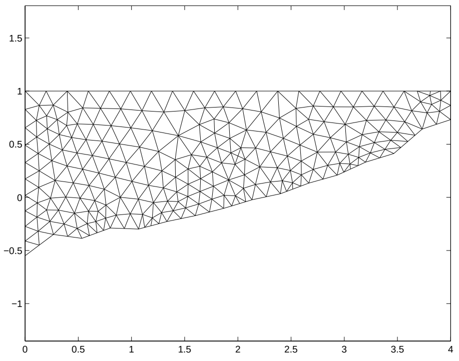

Figure 8: The shape after 8 iterations, modified gradient

The derivative becomes

$$\begin{aligned}
\partial_{P_i} S_R(p_h) &= \frac{\partial}{\partial P_i}\left[\alpha \sum_{i=2}^{N-1}\left(\frac{P_{i+1}-P_i}{\Delta x_i} - \frac{P_i - P_{i-1}}{\Delta x_{i-1}}\right)^2\right] \\
&= 2\alpha\Bigg[\left(\frac{P_{i+2}-P_{i+1}}{\Delta x_{i+1}} - \frac{P_{i+1}-P_i}{\Delta x_i}\right)\frac{1}{\Delta x_i} \\
&\quad + \left(\frac{P_{i+1}-P_i}{\Delta x_i} - \frac{P_i-P_{i-1}}{\Delta x_{i-1}}\right)\left(\frac{-1}{\Delta x_i} + \frac{-1}{\Delta x_{i-1}}\right) \\
&\quad + \left(\frac{P_i-P_{i-1}}{\Delta x_{i-1}} - \frac{P_{i-1}-P_{i-2}}{\Delta x_{i-2}}\right)\frac{1}{\Delta x_{i-1}}\Bigg]
\end{aligned}$$

with the modification for the derivatives with respect to $P_1,\ P_2,\ P_{N-1},\ P_N$ such that for $\partial_{P_1} S_R(p_h)$ only the first of the three terms remain, and for $\partial_{P_2} S_R(p_h)$ the third term goes away. Analogously, for $\partial_{P_{N-1}} S_R(p_h)$ the first term is not present and for $\partial_{P_N} S_R(p_h)$ only the third term is left. If the alternative discretization, $p_h = \sum_i e^{P_i}\psi_i$ is used the derivative becomes slightly different.

We try this for bilinear elements with 16 control nodes, using the projected gradient. The shape becomes smooth and we achieve a smaller displacement at the end. In figure 9 we see the shape after 35 iterations setting the penalty parameter $\alpha = 0.001$.

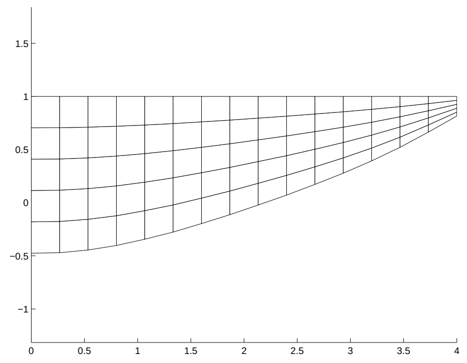

Figure 9: The shape with regularization parameter $\alpha = 0.001$

We compare different values of α in figure 10 and in table 7. The results are similar for triangular elements.

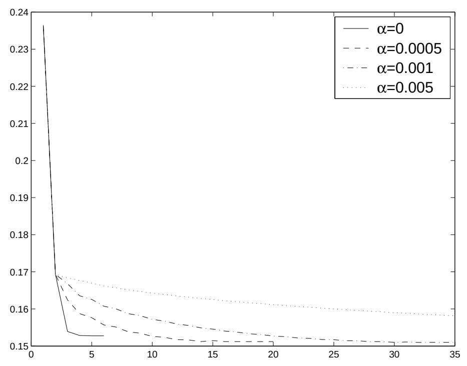

Figure 10: The displacement at the end as a function of iteration index for different regularization parameters α

We see that we find a better shape but we have slower convergence for increasing α. Note that during the first iterations the regularization has no influence.

This method removes the problem at the left end, but it does not explain why the phenomenon arises in the first place. We try to find an answer in the following subsections.

4.8 Using a finer state mesh

If the FE-mesh is modified so that there may be many state elements for each control element the oscillating behavior at the left end seems to decrease.

α	Iter for convergence	$U(L,0)$
0	5	0.1529
0.0005	17	0.1512
0.001	33	0.1510
0.005	61	0.1506

Table 7: Comparison for different α

Figure 11 is for instance the optimal shape when two state elements for each control element was used. We denote the number of state elements per

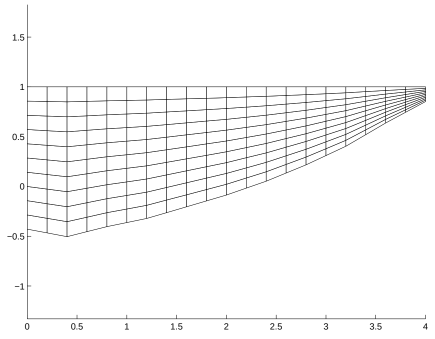

Figure 11: The optimal shape using a finer state mesh

control element d. In figures 12 and 13 are comparisons of the shape for $d = 1$, $d = 2$, $d = 4$ and $d = 8$.

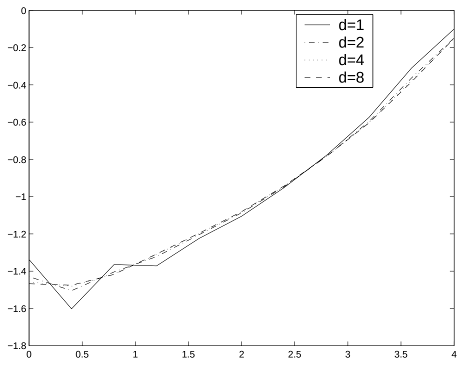

Figure 12: The shape with different partitions

Since the state mesh differs, the absolute value of the displacements are not good enough for comparison. Therefore the improvement in percentage is calculated and is together with the displacements collected in table 8. It is really hard to say if a smooth surface is in fact better based on the results here, because with a finer state mesh, the state solution becomes

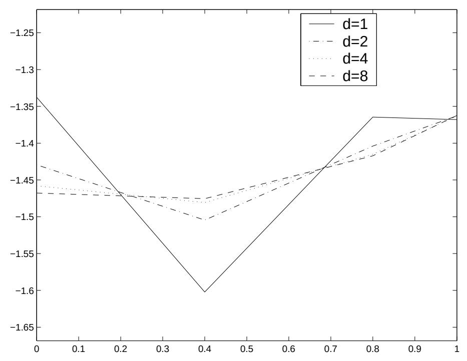

Figure 13: The shape with different partitions, zoomed in at the left end

more accurate, and since the continuous displacement is larger than the displacement of the FE-solution ("the FE-solution is stiffer"), a finer FE-mesh increases the displacement.

division	$U(L,0)$	improvement
1	0.1470	0.3603
2	0.1596	0.3398
4	0.1637	0.3342
8	0.1657	0.3301

Table 8: Results for Steepest descent, projected gradient, different meshes

In the following subsections we present possible explanations why a finer state mesh is useful.

4.9 Minimizing the energy

Up to now we have only considered minimization of the displacement. If instead the elastic energy of the console is to be minimized the objective functional becomes

$$\mathcal{I} = \frac{1}{2}||u||_E^2 + S(p) = \frac{1}{2}a(p;u,u) + S(p)$$

where $S(p)$ is a penalty functional. Then the Lagrangian in the discrete setting is

$$\mathcal{L} = \frac{1}{2}a(p_h;u_h,u_h) + S(p_h) + a(p_h;u_h,\lambda_h) - l(p_h;\lambda_h).$$

As above the u-derivative is to be zero, which gives the dual problem

$$\frac{1}{2}a(p_h;u_h,\varphi_i) + \frac{1}{2}a(p_h;\varphi_i,u_h) + a(p_h;\varphi_i,\lambda_h) = 0 \quad \forall \varphi_i \in V^h$$

so that by the symmetry of $a(p;\cdot,\cdot)$ the solution to the dual problem must be $\lambda_h = -u_h$. But the state equation gives

$$a(p_h; u_h, u_h) = l(p_h; u_h) \quad u_h \in V^h$$

so minimizing $a(p_h; u_h, u_h)$ is exactly the same as minimizing $l(p_h; u_h)$. If now the only load is a point load, the displacement u in the point where the point load is applied is minimized, providing a nice property: We minimize the total elastic energy by adding material (i.e increase stiffness) where the local elastic energy is high. More general, if a distributed load f is applied, the f-weighted displacement is minimized. In the 2D test problem (10) above a point load at the end is applied, and in that point the displacement is to be minimized, so in fact the elastic energy is minimized!

The derivative of the bilinear form with respect to the control was (14)

$$\partial_{P_i} a(p_h; u_h, \lambda_h) = \int_0^L [\sigma(u_h) : \epsilon(\lambda_h)]_{\Gamma_{p_h}} \psi_i \, dx.$$

According to the discussion above, the relationship between the state and the dual solutions was $u_h = -\lambda_h$. This gives

$$\partial_{P_i} a(p_h; u_h, \lambda_h) = -\int_0^L [\sigma(u_h) : \epsilon(u_h)]_{\Gamma_{p_h}} \psi_i \, dx.$$

This means that it is the elastic energy $\sigma(u_h) : \epsilon(u_h)$ on the boundary that dominates the shape derivative. This elastic energy is plotted as an average on each element in figure 14. The energy at the initial shape has three peaks, two at the left ends of the console and one where the force is applied. Notice the peak in the energy at the lower left end in figure 14. If the

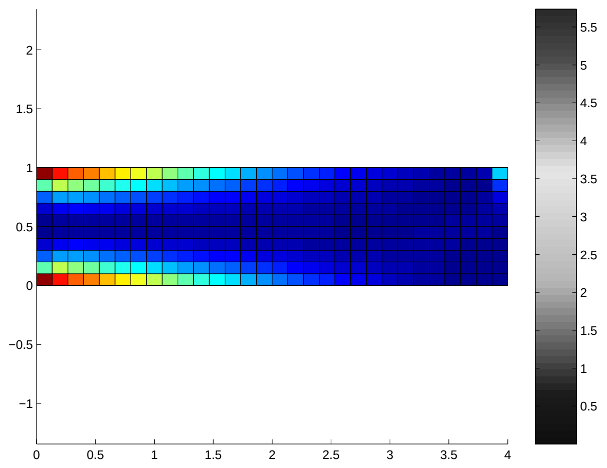

Figure 14: The energy distribution on the original shape

state mesh is not fine enough, the corresponding component in the shape derivative gets too small. If a coarse state mesh is used, then multiplying this component by two as suggested in the modified gradient above (subsection 4.6) compensates for this. Using the finer mesh also helps, as shown in the previous subsection. The elastic energy is also plotted for the converged

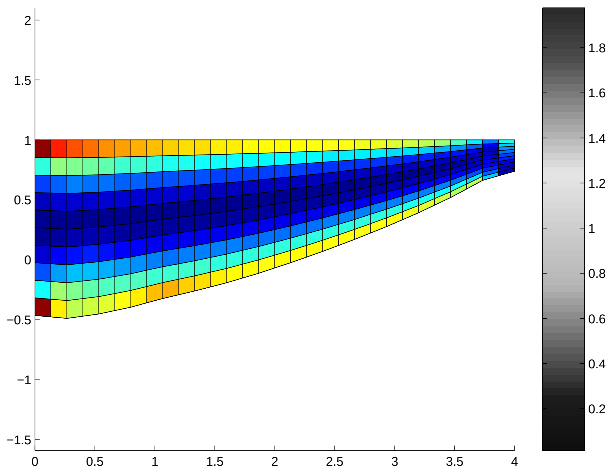

Figure 15: The distribution of energy on the optimal shape with 15 control elements, $d = 2$ and steepest descent without modification, 6 iterations

shape in figure 15. Note how the energy distribution is more smooth the optimal shape compared to the initial shape. The maximum value of the energy has decreased significantly. The energy distribution on the lower side is almost constant, which correspond to an almost constant derivative, which when the total area is kept constant (see subsection 3.3) correspond to a zero derivative, i.e. a stationary point.

If this really is the explanation, the local refinement would be sufficient and not a global one that is used above. This has not been tested for the bilinear case, because it would require a significant rewriting of the code. For the triangular elements a local refinement for the element close to the lower left end has been done. The final shapes for different number of refinement using 10 control nodes can be found in figure 16 and a comparison of the displacement $u_y(L, 0)$ at the end is found in figure 17. It seems that a refinement of the state mesh at the lower left does not improves the optima. This has also been tested for 6 control element, but no improvement in the shape was found there either. This is probably because the peak in energy at the left end is not very sharp. If the peak in energy would be the reason for the strange behavior at the left end, then the energy at the left end would have to be about twice as large as the energy at the second control node. Looking at the energy plots above figure 14 and 15 we see that the difference is much smaller, i.e. the peak is not sharp enough for the energy peak to be

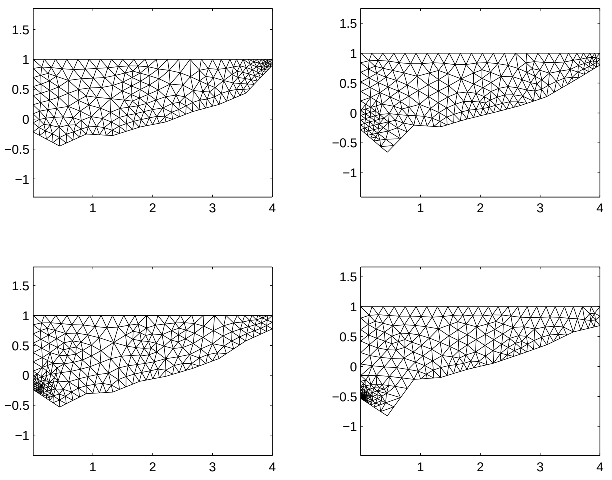
Figure 16: The optimal shape for zero, one two and four refinements, 10 control nodes

the whole explanation of the strange behavior at the left end.

4.10 Stability problems

Another possible explanation for the phenomenon at the left end is that it in fact is a stability problem, a so-called *checkerboard*-phenomenon. We see that the approximate solution p_h overestimates and underestimates the expected solution on every other node at the left end.

We have that the control p_h is piecewise linear. This means that for the shape derivative to be able to control p_h it also needs to be piecewise linear, and since the shape derivative is more or less the quantity $\sigma(u_h) : \epsilon(\lambda_h)$ it too has to be piecewise linear, but with a piecewise linear u_h and λ_h, $\sigma(u_h) : \epsilon(\lambda_h)$ becomes piecewise constant. This problem can be treated in two ways. First quadratic base functions for both the state u_h and dual λ_h and linear p_h (or alternatively piecewise linear for the state and dual and piecewise constant control p_h) may be used. Or secondly a much finer mesh for the state u_h (and the dual λ_h) can be used so that the piecewise constant $\sigma(u_h) : \epsilon(\lambda_h)$ becomes almost piecewise linear compared to the mesh for p_h.

We revisit the 1D test problem where piecewise constant basis functions for the control and piecewise linear basis functions for the state were used. With the same FE-mesh for both the control and the state the solution was stable. If a more coarse mesh for the state was chosen the control got a shape that can be interpreted as checkerboard, which seems reasonable because then the state space is not "rich" enough to stabilize the control. For 15 state elements and 60 control elements using steepest descent method the shape is plotted in figure 18.

Now returning to the 2D test problem with linear basis functions for the

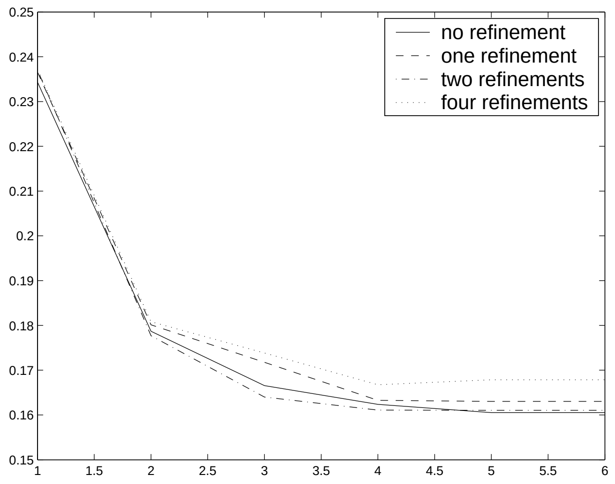

Figure 17: The displacement $u_y(L, 0)$ for zero, one, two and four refinements, 10 control nodes

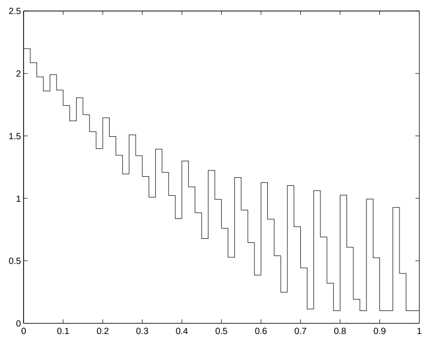

Figure 18: Optimal $p(x)$ (Steepest descent), 60 control and 16 states

state, dual and control. But with the FE-mesh for the control chosen to be much more coarse i.e. with a higher value of d, which seems to be a good cure for the problem encountered at the lower left part of the design.

Jog and Haber [8] studies the checkerboard phenomenon where the control and state have the same mesh on a 2D domain and a Newton-method is used. They find that only certain combinations of elements for state and control are stable, for instance a quadratic 8-node element for the state and a elementwise constant control is a stable combination, while a linear 4-node element for the state and elementwise constant control is unstable. Their results cannot be directly translated to this thesis, because here the state and control meshes are different.

4.11 Newton's method triangular elements

Here we use the method with the side conditions enforced via Lagrangian multipliers as described in subsection 4.2. We begin with 10 nodes for the control, using Newton's method as described straight on gives the shape in figure 19.

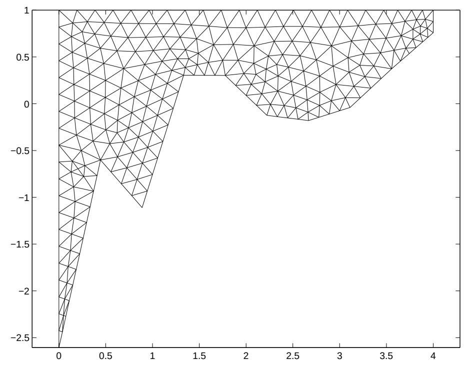

Figure 19: The shape after 12 iterations, Newton's method, undamped

During the iterations it seems that the Newton-steps taken are too long, resulting in an oscillating design path where the console is too thin in one step and in the next iteration it is too thick, and in the next iteration too thin again and so on. This might be compensated with the use of a damping, where the Newton-step in the control δp is multiplied with a small constant.

When using a damped Newton's method, with a damping of $1/2$, i.e the step in the control is $1/2$ of the usual Newton-step the results are found in table 9 and figure 20.

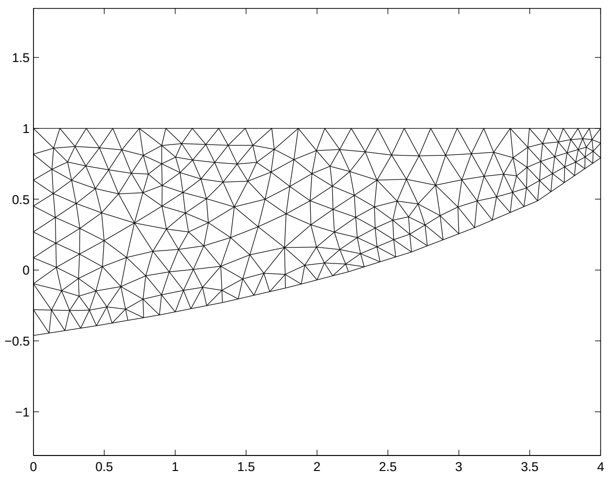

Figure 20: The shape after 50 iterations, Newton's method

Iter	Time	$\Delta U(L,0)$	$\|\|\Delta P_i\|\|$	$\|\|\partial_p \mathcal{L}\|\|$	$\|\|U - U^*\|\|$	$U(L,0)$	$U^*(L,0)$
0	0	-	-	-	-	0.2343	0.2343
1	5.054	2.552E-04	4.830E-03	2.702E-01	2.166E-03	0.2341	0.2343
2	23.398	3.810E-02	7.887E-01	2.670E-01	3.132E-01	0.1950	0.2331
3	33.756	2.449E-03	4.844E-01	1.339E-01	1.611E-02	0.1503	0.1528
4	41.875	4.173E-04	3.078E-01	1.212E-01	4.011E-03	0.1445	0.1449
5	50.172	-7.885E-05	2.367E-01	1.234E-01	7.361E-04	0.1463	0.1462
6	57.653	2.810E-05	1.305E-01	1.263E-01	4.276E-04	0.1480	0.1480
7	69.962	-1.286E-04	3.139E-02	1.267E-01	9.712E-04	0.1493	0.1491
8	79.769	2.474E-05	2.153E-02	1.276E-01	5.417E-04	0.1500	0.1500
9	89.091	-3.663E-05	1.844E-02	1.283E-01	3.462E-04	0.1503	0.1503
10	97.460	3.369E-05	1.542E-02	1.282E-01	3.378E-04	0.1504	0.1504
11	103.415	-1.006E-04	2.531E-02	1.286E-01	2.595E-04	0.1505	0.1505
12	111.176	9.564E-06	7.315E-02	1.288E-01	1.430E-04	0.1505	0.1505
13	119.451	4.861E-05	3.798E-02	1.281E-01	5.032E-04	0.1506	0.1506
14	127.621	-2.205E-05	2.403E-02	1.284E-01	2.539E-04	0.1505	0.1504
15	135.304	4.481E-06	1.725E-02	1.280E-01	4.433E-05	0.1504	0.1504
16	142.500	-1.738E-05	1.359E-02	1.284E-01	1.903E-04	0.1505	0.1505
...	...	...	...	...	...	...	...
50	412.003	1.269E-05	6.476E-03	1.275E-01	1.064E-04	0.1505	0.1506

Table 9: Results for Newton's method

We note that after about 10 iterations, newtons method does not come any closer and comes in a sort of "steady-state". We see that we have found a better shape using Newton's method compared to the steepest descent since the shape is smooth. When comparing to the results for the steepest descent method (with or without modified gradient) we must remember that here we have a more coarse grid since only 10 control nodes are used, so the numerical values for the displacement cannot be compared straight on. Note that there is no oscillating phenomenon at the left end. Although it seems that there is no problem at the left end we test global and local refinement of the state mesh. Local refinement at the left end gives after 50 newton-iterations using a damping of 1/2 gives the shape in figure 21. The shape for global refinement is also found in figure 21.

It seems that refinement induce some sort of instability. Similar phenomena occurs when 12 or more control nodes are used, for all tested dampings. Relating back to the discussion on stability in the previous subsection, we have by our refinement made the state space *more* capable to describe the piecewise linear control, so any instability phenomena should be reduced by refinement, which suggest that the stability discussion above is not the full explanation either.

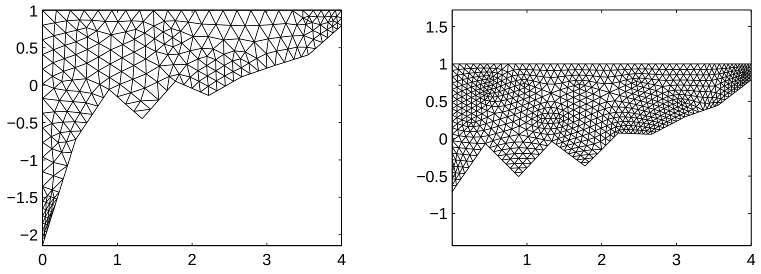

Figure 21: The shape after 50 iterations, local refinement (left) and global refinement (right)

4.12 A not well-defined minima

Another possibility for instability problem is that for this test problem, the minima is not well-defined. We do not achieve the typical quadratic convergence of the Newton method (we need to use damping). We note that for both the steepest descent method and for Newton's method, we do a number of iterations where a lot happens and we get close to the minima. Then the steepest descent method cannot find a step length that reduce the objective functional and Newton's method goes in circles. For a not well-defined minima a small change in the control does not result in a change in the objective functional, and therefore a phenomenon such as in figure 4 and 5 may arise. On the other hand, intuitively one feels that rather the minima should be quite well-defined, as for the 1D test problem.

5 Implementation aspects

The two test problems has been implemented in matlab. The code is quite long and is therefore not included in this thesis. To acquire the code for further testing, please contact the author.

The problems encountered during implementation are mainly due to the administration of the control p. Mostly this makes the assembling of the element contributions difficult. The element contributions themselves are quite similar to each other, so when the code for the standard element stiffness matrix is written the other matrices are easy to implement.

To solve the dual problem is in fact quite cheap. If the dual problem is independent of the state problem they can be solved simultaneously since they both have the same stiffness matrix. This gives a problem of the form $A[u\ \lambda] = [F\ G_{dual}]$, where A is the stiffness matrix and F and G_{dual} are the load vectors for the state and dual equations respectively. If the dual problem depends on the state equation then A may be LU-factorized to save time.

The method for calculating the shape derivative considered in this thesis is quite efficient, it requires first the solution of the dual problem (see above) and then the cost for the shape derivative itself is about the same as for an ordinary load vector. It is much more efficient than calculating the derivative by brute force, i.e. calculating $(\mathcal{I}(p + he_i, u(p + he_i)) - \mathcal{I}(p, u(p)))/h$ for all i.

6 General conclusions and other applications

In general the methodology outlined above seems to works fine, but evidently the instability problem need further attention. We see that we can take care of it by using a regularization or Newton's method (with few control nodes). Even though we are able to achieve a smooth shape, we have not found the explanation for why the oscillating phenomenon arises in the first place. We suggest three possibilities, a peak in the energy, a need for a rich state space and a not well-defined minima. It seems that of these explanations the not well-defined minima seems most likely.

There is an other issue that also need attention, how to deal with the side conditions and the constraints. Mainly three options are available, penalization, strong enforcement via some projection technique and Lagrangian multipliers. We are unable to draw any certain conclusions about which one is preferable since all methods works fine in some cases and not in other cases. One way to avoid side conditions may be a clever choice of parametrization, for instance using e^{P_i} instead of P_i (see subsection 4.2) to avoid negative P_i.

We see that there is a close connection between maximizing the stiffness and minimizing the elastic energy. Since the energy is a quite local property, a large optimization problem can be solved using adaptive techniques where the state and control meshes can be locally refined where the elastic energy is large and a more coarse mesh can be used where the elastic energy is small. This would bring down the computing time significantly.

We can think of another application. Suppose we have the 2D test problem as stated above, but with uniform thickness in the y-direction. We let the control $p(x, y)$ instead describe the stiffness of the material at every point, for instance the Youngs modulus of elasticity E, and we seek the control $p(x, y)$ such that the displacement is minimized while the total available material $\int_\Omega p(x, y)\, d\Omega$ is kept constant. A problem of this type is studied by Bengzon [2] with for example the application in the field of bone remodeling. Here there is a close connection to topology optimization where the setting is similar, but with a restriction that the control must be either 1 or 0. There are some methods to achieve this, for instance a penalty of the form $\alpha \int_\Omega p(1-p)\, d\Omega$, see Bendsøe [5] for an overview. There is a topology optimization method called Evolutionary Structural Optimization, ESO. It is a quite intuitive method; a loop starts where the state is computed and the stress on each element is calculated. Then the elements with the lowest stresses are removed and then loop starts again. If the criteria for removing an element is based on the elastic energy $\sigma(u) : \epsilon(u)$ we are quite close to the discussion of the elastic energy in the 2D test problem, subsection 4.9. An introduction to ESO can be found in Xie and Steven [13]. We conclude that the methodology presented in this thesis is quite general and has connections to size-, shape- and topology optimization, which shows the close connections between the three types of Structural Optimization.

7 Future work

Of course the oscillating phenomenon at the left end of the 2D test problem need much deeper studies than done here, foremost how well enrichment of the state space using a finer state mesh really works. Local refinement at the left end for the bilinear elements should also be tested to be certain that the peak in elastic energy is not the cause of the problem.

In order to determine when an algorithm has converged, a posteriori error estimates should be incorporated in this procedure. This might explain why Newton's method "goes into circles" close to the optima, the discretization error might be of the same magnitude as the Newton-step.

Additionally it would be interesting to study how an error in the control or the state results in an error in the objective functional. This may give an alternative view on why stability problems occur. Becker *et al* [1] studies a posteriori error estimates and adaptivity.

We have found a close connection between minimizing the energy and maximizing the stiffness. Perhaps there is a more general formulation for this, presumably using another energy norm, that will result in an simple formula for finding where something interesting happens.

Another approach to shape optimization problem is to study mappings where a reference mesh is mapped to a new domain and the problem is to find the mapping which makes the new domain optimal. Becker *et al* [1], Chenais *et al* [4] and Laumen [10] do their work using a mapping technique. The advantage is that the FE-mesh does not need to be updated during the design cycles. In practical use of FEM the mesh generation takes a considerable amount of time, thus a method that does not need a new mesh in each iteration is desirable.

Jog and Haber [8] gives a few suggestion on how to deal with the stability problem. They suggest among other things a regularization of the Lagrangian, which may be further investigated. Preliminary test shows that it works good. They also suggest a (patch) test to determine whether a combination of state and control elements is stable or not. This test could be performed for the test problems in this thesis too, and conform with our findings that the state space should be somewhat richer than the control space to obtain a stable optimization process.

Another aspect is that the state problem itself might not be determined, for instance, the load in the 2D-problem may be a random variable both in position and magnitude. Taking this into account may result in an other optimal shape. This can be further investigated. It might even be so that there is a connection between the a posteriori error estimations and this probabilistic approach to the optimization problem.

References

[1] R. Becker, H. Kapp, and R. Rannacher. Adaptive finite element methods for optimal control of partial differential equations: basic concept. *SIAM J. Control Optim.*, 39(1):113–132, 2000.

[2] F. Bengzon. (to appear). Master's thesis, CTH, 2002.

[3] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 1(1):1–55, 1995.

[4] D. Chenais, J. Monnier, and J. P. Vila. Shape optimal design problem with convective and radiative heat transfer: Analysis and implementation. *J. of Optim. Theory and Appl*, 110(1):75–117, 2001.

[5] M. P.B̃endsøe. *Optimization of Structural Topology, Shape, and Material.* Springer, 1995, ISBN 3-540-59057-9.

[6] K. Eriksson and M. G. Larson. Adaptive finite element approximation of the optimal variable thickness sheet problem. Article draft.

[7] P. Hansbo. Beyond the elements of finite elements: General principles for solid and fluid mechanics applications. Lecture notes.

[8] C. S. Jog and R. B. Haber. Stability of finite element models for distributed-parameter optimization and topology design. *Comput. Methods Appl. Mech. Engrg.*, 130:203–226, 1996.

[9] W. M. Lai, D. Rubin, and E. Krempl. *Introduction to Continuum Mechanics.* Butterworth-Heinemann, 1993, ISBN 0-7506-2894-4.

[10] M. Laumen. Newton's method for a class of optimal shape design problemes. *SIAM J. Optim.*, 10(2):503–533, 2000.

[11] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming.* McGraw-Hill, 1996, ISBN 0-07-114537-0.

[12] N. Ottosen and H. Petersson. *Introduction to the Finite Element Method.* Prentice Hall, 1992, ISBN 0-13-473877-2.

[13] Y. M. Xie and G. P. Steven. *Evolutionary Structural optimization.* Springer, 1997, ISBN 3-540-76153-5.

A A short summary of linear elasticity

This is a short description of Navier's elasticity equations used in the 2D test problem above. The equations are first stated in 3D and then we explain how the plain strain assumption reduces the problem to 2D. For the details, consult a standard textbook in solid science, here below we used Lai *et al* [9] and Hansbo [7]. Let the indices $i, j \in \{1, 2, 3\}$ represent the directions of coordinate axes x_i The stress σ is represented as a 3-by-3 matrix with the components σ_{ij}. The strain ϵ is also represented by a 3-by-3 matrix. The displacement u is represented by a 3-by-1 vector and the volume force f has also three components (represented in a vector). In the domain Ω, we assume force equilibrium, that is the stress σ and the volume force f are related by

$$-\nabla \cdot \sigma = f \qquad \text{in } \Omega$$

The relation between the stress and the strain is given by a constitutive law. For linear elasticity that relation is Hooke's law, which can be written as

$$\sigma = \lambda(\nabla \cdot u)I + 2\mu\epsilon \qquad \text{in } \Omega \tag{15}$$

where I is the identity matrix and λ and μ are Lamé constants, which can be expressed by Young's modulus of elasticity E and Poisson's ratio ν by

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

Assuming small strain, we get the relationship between the strain and the displacement as

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

Combining the above equations we get Navier's elasticity equation

$$(\lambda + \mu)\sum_j \frac{\partial^2 u_j}{\partial x_j \partial x_i} + \mu \sum_j \frac{\partial^2 u_i}{\partial x_i \partial x_j} = 0 \quad i = 1, 2, 3. \tag{16}$$

On a Neumann boundary $\partial\Omega_1$ with the normal n the normal component of the stress equals a stress vector g as $\sigma \cdot n = g$ on $\partial\Omega_1$. With a suitable test space V the weak form of (16) reads (c.f. Hansbo [7])

$$\int_\Omega \sigma(u) : \epsilon(v)\, d\Omega = \int_\Omega f \cdot v\, d\Omega + \int_{\Gamma_1} g \cdot v\, ds \qquad \forall v \in V \tag{17}$$

which in an abstract setting would read $a(u, v) = l(v) \;\; \forall v \in V$. Now, if a plane strain state is assumed, the components of the strain out of the plane are zero. If x_3 is the direction normal to this plane, then $\epsilon_{13} = \epsilon_{23} = \epsilon_{33} = 0$. Inserting this in the constitutive law (15) gives $\sigma_{13} = \sigma_{23} = 0$ and $\sigma_{33} = \lambda(\nabla \cdot u) \neq 0$. The fact that $\sigma_{33} \neq 0$ is really not needed since it in (17) is multiplied only with ϵ_{33} which is zero. Therefore when assuming plane strain the only change in the above equations is that now we only have two directions $i, j \in \{1, 2\}$.

B The derivative of the bilinear form with respect to p

Taking the derivative of the bilinear form with respect to the domain, i.e. finding $\partial_{P_i} a(p_h; u_h, \lambda_h)$ and $\partial^2_{P_i P_j} a(p_h; u_h, \lambda_h)$ is not trivial. For simplicity, consider a function $f(x_1, x_2)$ on a domain $\Omega(p_h)$ defined by $p_h = \sum_i P_i \psi_i$, where $\bar{x}_2$ is the x_2-coordinate of the lower boundary (see figure 22).

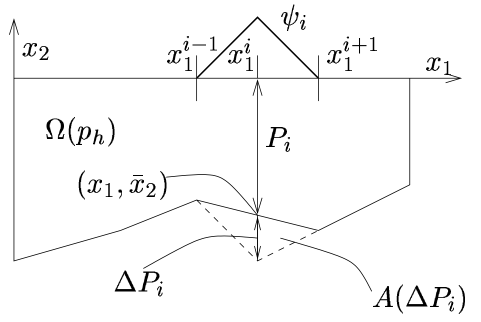

Figure 22: How a change in P_i changes the geometry

A small increase ΔP_i in the node value P_i extends the domain by the small area $A(\Delta P_i)$. The derivative $\partial_{P_i}\left(\int_{\Omega(p_h)} f(x_1, x_2)\, dx_1\, dx_2\right)$ can then be derived by using the limes definition of the derivative.

$$\partial_{P_i}\left(\int_{\Omega(p_h)} f(x_1, x_2)\, dx_1\, dx_2\right) = \lim_{\Delta P_i \to 0} \frac{\int_{A(\Delta P_i)} f(x_1, x_2)\, dx_1\, dx_2}{\Delta P_i}.$$

The Taylor expansion of f in the x_2-direction from $\bar{x}_2$ gives

$$\begin{aligned}
\frac{\int_{A(\Delta P_i)} f(x_1, x_2)\, dx_1\, dx_2}{\Delta P_i} &= \frac{\int_{A(\Delta P_i)} f(x_1, \bar{x}_2) + f'_{x_2}(x_1, \bar{x}_2)\Delta x_2 + \mathcal{O}(\Delta x_2^2)\, dx_1\, dx_2}{\Delta P_i} \\
&= \frac{\int_0^L f(x_1, \bar{x}_2)\Delta P_i \psi_i + \frac{1}{2} f'_{x_2}(x_1, \bar{x}_2)\Delta P_i^2 \psi_i^2 + \mathcal{O}(\Delta P_i^3)\, dx_1}{\Delta P_i} \\
&= \int_0^L f(x_1, \bar{x}_2)\psi_i\, dx_1 + \Delta P_i \int_0^L \frac{1}{2} f'_{x_2}(x_1, \bar{x}_2)\psi_i^2\, dx_1 \\
&+ \int_0^L \mathcal{O}(\Delta P_i^2)\, dx_1
\end{aligned}$$

where we used that $\Delta x_2 = -\Delta P_i \psi_i$. And letting $\Delta P_i \to 0$ gives the derivative

$$\partial_{P_i}\left(\int_{\Omega(p_h)} f(x_1, x_2)\, dx_1\, dx_2\right) = \int_0^L f(x_1, \bar{x}_2)\psi_i\, dx_1.$$

A new small change in P_j this time changes $\bar{x}_2$

$$\partial_{P_j}\left(\int_0^L f(x_1, \bar{x}_2)\psi_i\, dx_1\right) = \lim_{\Delta P_j \to 0} \frac{\int_0^L f(x_1, \bar{x}_2 - \Delta P_j \psi_j)\psi_i\, dx_1 - \int_0^L f(x_1, \bar{x}_2)\psi_i\, dx_1}{\Delta P_j}.$$

The Taylor expansion of the first term is

$$\int_0^L f(x_1, \bar{x}_2 - \Delta P_j \psi_j)\psi_i \, dx_1$$

$$= \int_0^L f(x_1, \bar{x}_2)\psi_i - f'_{x_2}(x_1, \bar{x}_2)\Delta P_j \psi_j \psi_i dx_1 + \mathcal{O}(\Delta P_j^2) \, dx_1.$$

Taking the limit results in

$$\partial_{P_j} \left(\int_0^L f(x_1, \bar{x}_2)\psi_i \, dx_1 \right) = - \int_0^L f'_{x_2}(x_1, \bar{x}_2)\psi_j \psi_i \, dx_1.$$

With the bilinear form $a(p_h; u_h, \lambda_h) = \int_{\Omega(p_h)} \sigma(u_h) : \epsilon(\lambda_h) \, d\Omega$ the desired derivatives become

$$\partial_{P_i} a(p_h; u_h, \lambda_h) = \int_0^L \left[\sigma(u_h) : \epsilon(\lambda_h)\right]_{\bar{x}_2} \psi_i \, dx_1$$

$$\partial_{P_i P_j} a(p_h; u_h, \lambda_h) = - \int_0^L \left[\partial_{x_2} \left(\sigma(u_h) : \epsilon(\lambda_h)\right)\right]_{\bar{x}_2} \psi_i \psi_j \, dx_1.$$

If the discretization instead is $p_h = \sum_i e^{P_i}\psi_i$ the derivations are the same, but the height of the extra area A because of ΔP_i is instead $\Delta Q_i = e^{P_i + \Delta P_i} - e^{P_i}$, which in the calculations above gives $\Delta x_2 = -\Delta Q_i \psi_i$ in the numerator. Taking the limes this time gives

$$\begin{aligned} \partial_{P_i} a(p_h; u_h, \lambda_h) &= \lim_{\Delta P_j \to 0} \frac{\Delta Q_i}{\Delta P_i} \int_0^L \sigma(u_h) : \epsilon(\lambda_h)\psi_i \, dx_1 + \frac{\mathcal{O}(\Delta Q_i^2)}{\Delta P_i} \\ &= e^{P_i} \int_0^L \left[\sigma(u_h) : \epsilon(\lambda_h)\right]_{\bar{x}_2} \psi_i \, dx_1. \end{aligned}$$

Similarly

$$\partial_{P_i P_j} a(p_h; u_h, \lambda_h) = -e^{P_i} e^{P_j} \int_0^L \left[\partial_{x_2} \left(\sigma(u_h) : \epsilon(\lambda_h)\right)\right]_{\bar{x}_2} \psi_i \psi_j \, dx_1.$$

What is then $\partial_{x_2} \left(\sigma(u_h) : \epsilon(\lambda_h)\right)$? First we use the product rule

$$\partial_{x_2} \left(\sigma(u_h) : \epsilon(\lambda_h)\right) = \left(\partial_{x_2}\sigma(u_h)\right) : \epsilon(\lambda_h) + \sigma(u_h) : \left(\partial_{x_2}\epsilon(\lambda_h)\right).$$

Using Hooke's law to get the derivative of σ_{ij} $(i, j \in \{1, 2\})$

$$\partial_{x_2}\sigma_{ij} = \hat{\lambda} \left(\frac{\partial \epsilon_{11}}{\partial x_2} + \frac{\partial \epsilon_{22}}{\partial x_2} \right) + 2\hat{\mu} \frac{\partial \epsilon_{ij}}{\partial x_2}$$

where $\hat{\lambda}$ och $\hat{\mu}$ are the Lamé constants. Using the definition of strain to get

$$\frac{\partial \epsilon_{ij}}{\partial x_2} = \frac{1}{2} \left(\frac{\partial^2 u_i}{\partial x_2 \partial x_j} + \frac{\partial^2 u_j}{\partial x_2 \partial x_i} \right)$$

where the second derivatives of the displacements are needed. With the basis functions φ_k they are (for the dual λ_h they are similar)

$$\frac{\partial^2 u_i}{\partial x_2^2} = \sum_k U_k^i \frac{\partial^2 \varphi_k}{\partial x_2^2}$$

$$\frac{\partial^2 u_i}{\partial x_2 \partial x_1} = \sum_k U_k^i \frac{\partial^2 \varphi_k}{\partial x_2 \partial x_1}.$$

Using linear triangular element the basis functions according to Ottosen and Petersson [12] can be written as $\varphi = \alpha_1 + \alpha_2 x_1 + \alpha_3 x_2$. This means that for linear triangular elements both second derivatives becomes zero. For bilinear elements the basis functions can be written as $\varphi = \alpha_1 + \alpha_2 x_1 + \alpha_3 x_2 + \alpha_4 x_1 x_2$. Here the double x_2-derivative is zero while the mixed second derivative is α_4. According to [12] the **C**-method for determination of the basis function states that for an element where (x_1^i, x_2^i) are the coordinates for the element nodes the following relation is true

$$\begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 \end{bmatrix} \mathbf{C}^{-1}$$

where

$$\mathbf{C} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & x_1^1 x_2^1 \\ 1 & x_1^2 & x_2^2 & x_1^2 x_2^2 \\ 1 & x_1^3 & x_2^3 & x_1^3 x_2^3 \\ 1 & x_1^4 & x_2^4 & x_1^4 x_2^4 \end{bmatrix}.$$

The desired α_4 (one for each basis function on the element) is given as the last row of $\mathbf{C}^{-1}$. Thus we have all the needed ingredients to calculate $\partial_{x_2} \left(\sigma(u_h) : \epsilon(\lambda_h) \right)$.

C A general description of SQP

This section is a general description of SQP, Sequential Quadratic Programming and can be found in Boggs [3] or Nash & Sofer [11]. SQP is a general method for solving minimization problems of the type

$$\min_{\mathbf{x}} f(\mathbf{x})$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0}.$$

The minimization problem is solved iteratively by taking a step $\mathbf{d_x} = \mathbf{x} - \mathbf{x^k}$. This step is chosen as the minima of a simpler minimization problem. For this simpler problem to be easy, but yet a good description of the original problem a second order Taylor approximation in $\mathbf{x}$ of the objective function and a first order Taylor approximation of the side conditions is used. This gives a quadratic minimization problem

$$\min_{\mathbf{d_x}} \quad \mathbf{r^k d_x} + \frac{\mathbf{1}}{\mathbf{2}} \mathbf{d_x^T B_k d_x}$$

$$\nabla \mathbf{h}(\mathbf{x^k})\mathbf{d_x} + \mathbf{h}(\mathbf{x^k}) = \mathbf{0}$$

$$\nabla \mathbf{g}(\mathbf{x^k})\mathbf{d_x} + \mathbf{g}(\mathbf{x^k}) \leq \mathbf{0}$$

Which with $\mathbf{r^k}= \nabla f$ and $B_k = \nabla^2 f$ gives the desired step $\mathbf{d_x}$. Other choices of the vector $\mathbf{r^k}$ and the matrix B_k are possible for obtaining better numerical characteristics or to have a better global convergence. This is not studied here. The reason for approximation to a quadratic subproblem is that an efficient solver for this problem can be constructed, and thereby making SQP efficient for many different problems. For the special case when only equality constrains are present the quadratic subproblem can be solved very easily. We introduce the Lagrangian multiplier $\mathbf{u}$ with the step $\mathbf{d_u}$ for the equality constraint to get the Lagrangian

$$\mathcal{L} = \nabla f(\mathbf{x_k})^{\mathbf{T}} \mathbf{d_x} + \frac{\mathbf{1}}{\mathbf{2}} \mathbf{d_x} \nabla^{\mathbf{2}} \mathbf{f}(\mathbf{x_k}) \mathbf{d_x} + \left(\nabla \mathbf{h}(\mathbf{x^k})\mathbf{d_x} + \mathbf{h}(\mathbf{x^k}) \right) \mathbf{d_u}.$$

We find a stationary point for $\mathcal{L}$ by solving the linear system

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}(\mathbf{x^k}, \mathbf{u^k}) & \nabla \mathbf{h}(\mathbf{x^k}) \\ \nabla \mathbf{h}(\mathbf{x^k})^{\mathbf{T}} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d_x} \\ \mathbf{d_u} \end{bmatrix} = \begin{bmatrix} -\nabla_x \mathcal{L}(\mathbf{x^k}, \mathbf{u^k}) \\ -\mathbf{h}(\mathbf{x^k}) \end{bmatrix}$$

which gives an update for $\mathbf{x}$ and $\mathbf{u}$. Summing up, when only equality constraints are present an iteration in SQP is done by solving a linear system as in the usual Newton's method.