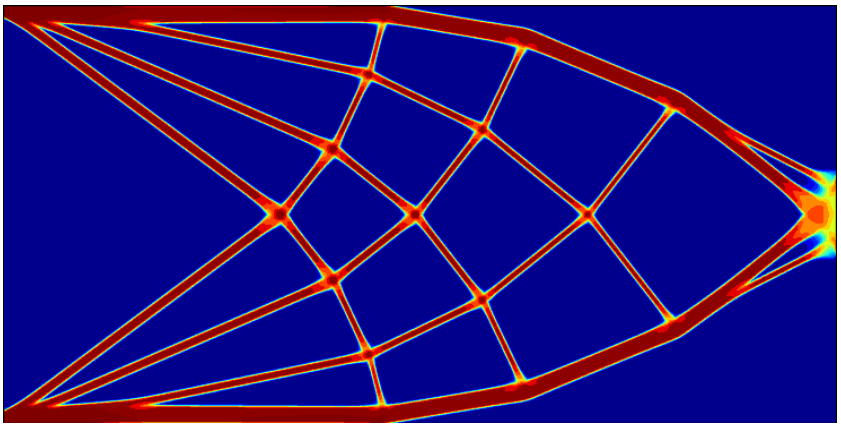
Programmier Praktikum WS2020/2021

Shape optimization with FEniCS

Gemeinsame Dokumentation



Dozent: Dr. Martin Lenz

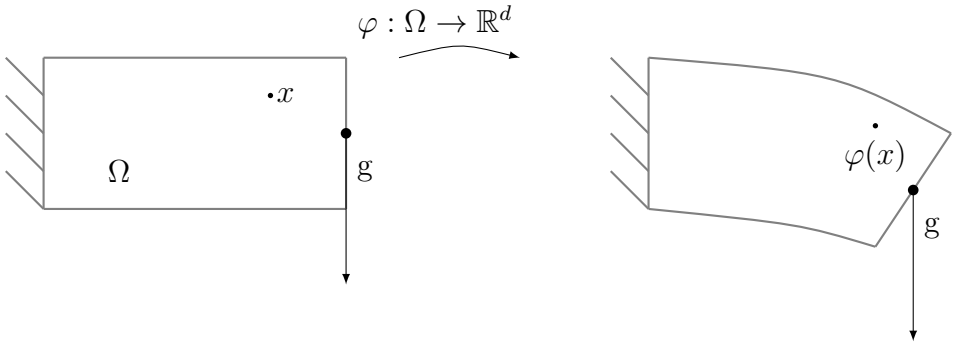
Bonn, am 31. Oktober 2020

1 28.10.2020 - Einführung in die Form Optimierung

1.1 Linearisierte Elastizität

Wir betrachten im Folgenden eine Deformation $\varphi : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ und definieren die Verschiebung (engl. displacement) $u : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ durch:

$$\varphi(x) = x + u(x).$$



Für kleine Verschiebungen lässt sich die Verzerrung (o.a. Dehnung, also lokale Längenänderungen) durch den linearisierten Dehnungstensor (strain tensor)

$$\varepsilon(u) = \frac{\nabla u + \nabla u^T}{2}$$

beschreiben. Die resultierende Spannung (stress) ist dann gegeben durch

$$\sigma(u) = \lambda \operatorname{div} u \mathbb{1} + \mu \varepsilon(u).$$

λ und μ sind materialabhängige Konstanten (Lamé-Parameter). Physikalische Deformationen minimieren dann Energien der Form:

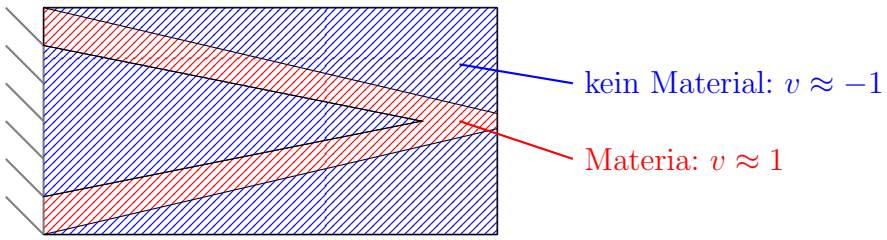
$$E(u) = \underbrace{\int_\Omega \sigma(u) : \varepsilon(u)\, dx}_{\text{gespeicherte Elastische Energie}} - \underbrace{\int_{\partial\Omega} g \cdot u\, ds}_{\text{Nachgiebigkeit}}.$$

Die gespeicherte elastische Energie (stored elastic energy) ist bei kleinen Verschiebungen klein und wächst mit höheren Spannungen und Dehnungen. g beschreibt am Rand wirkende äußere Kräfte. Die Nachgiebigkeit (compliance) wird klein, falls die Verschiebung sich an die Kräfte anpasst. Damit das für das Funktional ein Minimierer existiert und man diesen berechnen kann, müssen in diesem Fall noch Dirichlet Randwerte ($u = 0$ am linken Rand) gesetzt werden.

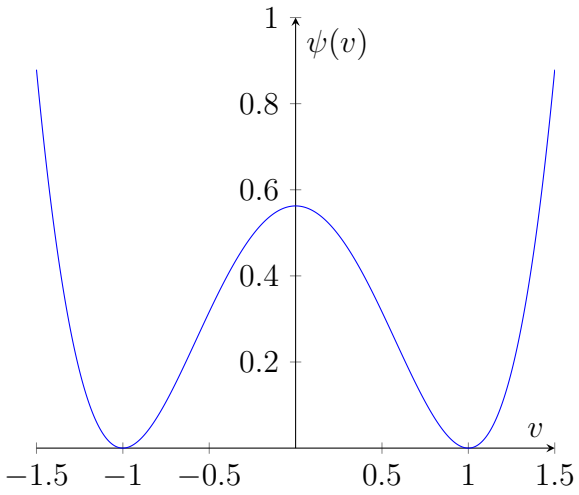
1.2 Phasenfelder

Um Formen und Geometrien mathematisch zu beschreiben gibt es mehrere Möglichkeiten. Wir werden zunächst sogenannte Phasenfelder (phase fields) benutzen. Das sind

Funktionen $v : \Omega \to \mathbb{R}$ mit $v(x) \approx 1$ für den Teil von Ω, der mit Material gefüllt ist und $v(x) \approx -1$ ansonsten. Dazwischen gibt es einen schmalen, glatten Übergang.



Um diese Eigenschaften zu erzwingen, benutzt man ein double-well-potential (zu deutsch vielleicht zwei Mulden Potential) $\psi : \mathbb{R} \to \mathbb{R}$ mit $\psi(v) = \frac{9}{16}\left(v^2 - 1\right)^2$:



Zusammen mit einer entsprechenden charakteristischen Funktion $\chi : \mathbb{R} \to \mathbb{R}$, $\chi(v) = \frac{1}{4}\left(v + 1\right)^2$ lässt sich ein Strafterm $P(v)$ einführen

$$P^{\varepsilon}(v) = \int_{\Omega} \alpha \underbrace{\frac{1}{2}\left(\varepsilon |\nabla v|^2 + \frac{1}{\varepsilon}\psi(v)\right)}_{\xrightarrow{\varepsilon \to 0} \text{Kantenlänge/Oberfläche}} + \underbrace{\beta\chi(v)}_{\text{Volumen}} \, dx$$

der dafür sorgt, dass die Oberfläche und das Volumen der beschriebenen Form klein bleiben. α und β sind hier Konstanten, die gewählt werden können, um den Oberflächenterm und den Volumenterm zu gewichten.
Für so beschriebene Flächen, lässt sich die gespeicherte elastische Energie für eine Verschiebung u ausdrücken durch:

$$E(u, v) = \int_{\Omega} ((1 - \delta)\chi(v) + \delta)\sigma(u) : \varepsilon(u) \, dx \, .$$

Für numerische Stabilität wählt man zumeist ein hard-soft-model, das heißt die Zwischenräume werden behandelt, als wären sie mit sehr weichem Material gefüllt.

1.3 Formoptimierung

Je nachdem, welches Ziel die Formoptimierung haben soll, kann nun ein sogenanntes Kostenfunktional (cost functional) $J(u, v)$ aufgestellt werden. Soll die Form z.B. so gewählt werde, dass die Verformung unter einwirkenden Kräften minimal bleibt, wählt man:

$$J(u, v) = P(v) + \int_{\partial\Omega} g \cdot u \, ds$$

Hier muss darauf geachtet werden, dass $v(x) \approx 1$, falls $g(x) \neq 0$ gilt. Das stellt sicher, dass an den Stellen wo Kräfte angreifen, Material vorhanden ist. Das Ziel der Formoptimierung ist nun:

$$\begin{aligned} &\text{Minimiere } J(u, v) \text{ bezüglich } v, \\ &\quad \text{wobei} u \quad E(u, v) \text{ minimiert} \end{aligned}$$

Sei nun $u(v)$ der Minimierer von $E(\cdot, v)$ für festes v. Wir setzten nun

$$\widehat{J}(v) = J(u(v), v).$$

Um die Ableitung von $\widehat{J}(v)$ nach v zu berechnen, benutzen wir die Kettenregel und erhalten

$$\widehat{J}_{,v}(v) = J_{,u}(u(v), v) \underbrace{u_{,v}(v)}_{?} + J_{,v}(u(v), v)$$

Da die Abhängigkeit u von v nur über ein Minimierungsproblem gegeben ist, ist unklar, wie die Ableitung $u_{,v}(v)$ zu berechnen ist. Deshalb wählt man (wie bei anderen Optimierungsproblemen mit Nebenbedingungn) den Ansatz der Lagrangemultiplikatoren und setzen

$$L(u, v, p) = J(u, v) + E_{,u}(u, v)p.$$

Notwendige Bedingungen zu Minimierung der Lagrange Funktion sind dann

$$\begin{aligned} &(1) \quad \partial_p L = E_{,u}(u, v) = 0 && \text{-> das sichert unser Nebenbedingung} \\ &(2) \quad \partial_u L = J_{,u}(u, v) + E_{,uu}(u, v)p = 0 && \text{-> lineare Gleichung, definiert } p \\ &(3) \quad \partial_v L = \underbrace{J_{,v}(u, v) + e_{,uv}(u, v)p}_{=\widehat{J}_{,v} \text{ Form-Gradient, s.u.}} = 0 \end{aligned}$$

Die dritte Gleichung liefert uns nach folgender Rechnung den Form-Gradienten:

$$\begin{aligned} \widehat{J}_{,v}(v) &= J_{,u}(u(v), v)u_{,v}(v) + J_{,v}(u(v), v) \\ &= -E_{,uu}(u(v), v)(p, u_{,v}(v)) + J_{,v}(u(v), v) && \text{nach (2)} \\ &= -E_{,uu}(u(v), v)(u_{,v}(v), p) + J_{,v}(u(v), v) && \text{mit Regularitätsannahmen} \\ &= E_{,uv}(u(v), v)p + J_{,v}(u(v), v) \end{aligned}$$

wobei der letzte Schritt aus Differentiation der notwendigen Bedingung nach v folgt:

$$\begin{gathered} E_{,u}(u(v), v) = 0 \\ \downarrow \partial_v \\ E_{,uu}(u(v), v) u_{,v}(v) + E_{,uv}(u(v), v) = 0 \end{gathered}$$

Damit haben wir eine Darstellung des Formgradienten $\widehat{J}_{,v}$, die sich ohne $u_{,v}$ berechnen lässt.

1.4 Was ist FEniCS?

FEniCS ist eine Open-Source-Platform zur numerischen Lösung von Partiellen Differentialgleichungen mit Python. Wie Python ist FEniCS eine höhere Programmiersprache (also abstrakter und weniger Maschinennah als z.B. C) und erlaubt einen knappen und sehr natürlichen Programmierstil. FEniCS benutzt effiziente lineare Algebra Bibliotheken und erlaubt automatisierte Parallelisierung, ist also trotzdem recht effizient.

1.5 Organisatorisches

- Die Programmierübungen dürfen in Gruppen mit bis zu 3 Studierenden abgeben werden
- Im Laufe des Praktikums wird dieses Latex-Dokument als gemeinsames Skript geführt und Inhalte, Präsentationen von Ergebnissen und Diskussionen des Praktikums festgehalten.
- für die Notenvergabe wird die „Projektarbeit und Präsentation“ bewertet, also insbesondere:
  - die Abgabe der Programmierübungen
  - die Beteiligung in Diskussionen
  - die Präsentation von Resultaten im Plenum (Jede Aufgabe wird von einer Gruppe vorgestellt)
  - der Beitrag zum gemeinsamen Skript.

1.6 Arbeitsumgebungen und erste Aufgabe

- FEniCS-Programme lassen sich in folgenden Arbeitsumgebungen ausführen:
  - Anaconda (Python Distribution für wissenschaftliches Rechnen, kein FEniCS Paket für Windwos verfügbar)
  - Docker (Software für Anwendungen in Containern)
  - Ubuntu Pakete

- Als Editor kann ein beliebiger Code Editor benutzt werden (z.B. Visual Studio Code)
- Zur Versionsverwaltung und zum Austausch von Code verwenden wir git (für die Abgabe der Übungsaufgaben GitHub Classroom)
- Zur Visualisierung von Resultaten eignet sich Paraview

Die ersten Aufgaben:

1. Installieren Sie eine Arbeitsumgebung auf Ihrem PC.
2. Führen Sie das Beispiel Programm von GitHub aus.
3. Laden Sie ein ParaView Bild des Ergebnis auf GitHub hoch.