Vienna University of Technology

Computational Mathematics - Seminary

Institute of Analysis and Scientific Computing

Shape Optimization in NGSolve

Authors:
Camilo Tello Fachin
12127084

Paul Genest
12131124

Supervisor:
Dr. Kevin Sturm

July 5, 2022

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Zusammenfassung

Zusammenfassung in Deutsch!

Abstract

Abstract in English!

Contents

1 Introduction 2

2 The Stokes Equations in NGSolve 3

3 Shape Optimization 4

4 Lagrangian and its Side Constraints 5

5 Shape Derrivative 6

6 Deformation 7

7 Iteration 8

8 Results and Conclusion 9

References 10

Appendices 11

A Python Code Listing 11

B XMI Code Listing 12

C MATLAB Code Listing 13

To Do's

- What's still to you do make your supervisor/prof happy?

1 Introduction

i want to cite [3] and also [1] and also [7] and also [2], A figure example and a text where I refer to figure 1 below! Additionally I need other citations like [5] as well as [4] and [6] yes yes

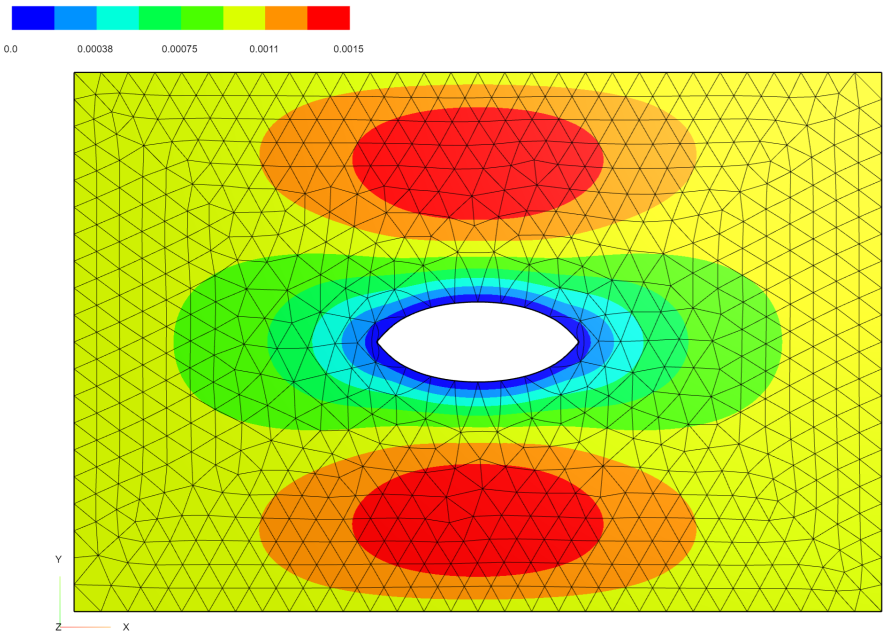

Figure 1: Velocity magnitude of Stokes flow after shape optimization

2 The Stokes Equations in NGSolve

The Stokes Equation, a linear partial differential equation, can describe a stationary incompressible Newtonian fluid with high viscosities and low Reynolds numbers. It is defined below in equation 1:

$$\begin{aligned} -\mu \Delta u + \nabla p &= f \\ \operatorname{div} u &= 0 \end{aligned} \tag{1}$$

Where $\mu \in \mathbb{R}$ is the viscosity constant and f the data (e.g. source). The problem yields the vectorial velocity field $u : \Omega \to \mathbb{R}^d$ and the scalar pressure field $p : \Omega \to \mathbb{R}$. In order to solve the Stokes equation with the Finite Element Method in NGSolve, it needs to be transformed to the weak formulation, where the solutions u and p are linear combinations of basis functions in a Sobolev space. See Faustmann[5] Chapter 3 for further elaborations on Sobolev spaces. The weak formulation can be derrived by multipling the now called trial-functions u and p with test-functions v and q, perform transformations and integrate them. The test-functions have to fulfil certain conditions to permit the transformations in order to arrive at a weak problem with linear convergence rates, see Faustmann[5]:

Find $u \in [H_0^1(\Omega)]^d$ and $p \in L^2(\Omega)$ such that

$$\begin{aligned} \int_\Omega \nabla u : \nabla v \, dx + \int_\Omega \operatorname{div}(v) p \, dx &= \int_\Omega f v \, dx \quad && \forall v \in [H_0^1(\Omega)]^d \\ \int_\Omega \operatorname{div}(u) q \, dx &= 0 && \forall q \in L^2(\Omega) \end{aligned} \tag{2}$$

The

3 Shape Optimization

4 Lagrangian and its Side Constraints

5 Shape Derrivative

6 Deformation

7 Iteration

8 Results and Conclusion

References

[1] V. Schulz and M. Siebenborn, "Computational comparison of surface metrics for PDE constrained Shape Optimization," *Computational Methods in Applied Mathematics*, vol. 16, Sep. 2015. DOI: https://doi.org/10.1515/cmam-2016-0009.

[2] J. Iglesias, K. Sturm, and F. Wechsung, "Two-Dimensional Shape Optimization with Nearly Conformal Transformations," *SIAM Journal on Scientific Computing*, vol. 40, A3807–A3830, Jan. 2018. DOI: https://doi.org/10.1137/17M1152711.

[3] P. Gangl, K. Sturm, M. Neunteufel, and J. Schöberl, "Fully and Semi-automated Shape Differentiation in NGSolve," *Structural and multidisciplinary optimization*, vol. 63, no. 3, pp. 1579–1607, 2021. DOI: https://doi.org/10.1007/s00158-020-02742-w.

[4] M. Faustmann, "Lecture notes for Applied Mathematics Foundations - TU Vienna ASC," Feb. 2022.

[5] M. Faustmann and J. Schoeberl, "Lecture notes for Numerical Methods for PDE's - TU Vienna ASC," Jun. 2022.

[6] J. M. Melenk, "Lecture notes for Numerical Computation - TU Vienna ASC," Feb. 2022.

[7] K. Sturm, "Lecture notes for PDE constrained Optimization - TU Vienna ASC," Jun. 2022.

A Python Code Listing

Here is an example of a python listing, you can change appearance of comments, strings, numbering, known commands and variables in the package settings in packages.tex. You can obviously use the listings environment in the rest of the document. The same procedure applies for listings in other languages.

Python Listing Title

```
1  # Python Script, API Version = V18
2
3  import math
4
5  #    DELETE EVERYTHING ------------------------------
6
7  ClearAll()
8
9  #    PARAMETERS -------------------------------------
10
11 w = float(Parameters.w)      # side length of one element or half of a unit cell
12 e = float(Parameters.e)      # rectangle ratio e
13 b = w/(1+e)
14 rho = float(Parameters.rho) # relative density
15 f = float(Parameters.f)      # number of layers = folds+1
16 h = 2*w/f           # layer height = size of a unit cell divided by the number of layers
17 f = int(Parameters.f)
18
19 # Calculation of wall thickness t
20 t1 = ((math.sqrt(1-rho)+1)*math.sqrt(2)*w)/2
21 t2 = -((math.sqrt(1-rho)-1)*math.sqrt(2)*w)/2
22 if t1 <t2:
23   t=t1
24 else:
25    t=t2
26
27 # auxiliary variable to build up rectangle
28 m = math.sqrt(pow(t,2)*2)/2
```

B XMI Code Listing

Here is an example for XML code listing.

XML Listing Title

```
1 <extension version="1" name="EnergyIntegral" loadasdefault="True">
2   <guid shortid="EnergyIntegral">8005c624-8869-4c74-b32b-97ac59c200b2</guid>
3   <script src="energy_integral.py" />
4   <interface context="Mechanical">
```

C MATLAB Code Listing

Here is an example for MATLAB code listing

MATLAB Listing Title

```
%% Linear model Poly44 from MATLAB Curve Fit App:

%Polynomial Coefficients (with 95\% confidence bounds):
       p00 =       13.79;  %(13.22, 14.36)
       p10 =      −2.897;  %(−3.454, −2.34)
       p01 =       3.752;  %(3.163, 4.34)
       p20 =       3.279;  %(2.231, 4.327)
       p11 =      0.5404;  %(−0.2001, 1.281)
       p02 =      0.8638;  %(−0.4624, 2.19)
       p30 =       0.299;  %(0.01281, 0.5851)
       p21 =     −0.5091;  %(−0.7299, −0.2884)
       p12 =      0.4973;  %(0.2716, 0.7229)
       p03 =      0.3595;  %(0.04484, 0.6741)
       p40 =     −0.8495;  %(−1.291, −0.4084)
       p31 =   −0.02258;  %(−0.3136, 0.2685)
       p22 =     −0.2819;  %(−0.5502, −0.01351)
       p13 =      0.2674;  %(−0.05265, 0.5874)
       p04 =      0.2019;  %(−0.3968, 0.8006)

    f(x,y) = p00 + p10*x + p01*y + p20*x^2 + p11*x*y + p02*y^2 + p30*x^3 + p21*x^2*y
    + p12*x*y^2 + p03*y^3 + p40*x^4 + p31*x^3*y + p22*x^2*y^2
    + p13*x*y^3 + p04*y^4

  %Goodness of fit:
  %SSE: 3.189
  %R−square: 0.9949
  %Adjusted R−square: 0.9902
  %RMSE: 0.4611
```