

VIENNA UNIVERSITY OF TECHNOLOGY

COMPUTATIONAL MATHEMATICS - SEMINARY

INSTITUTE OF ANALYSIS AND SCIENTIFIC COMPUTING

PDE Constrained Shape Optimization

Authors:

Camilo TELLO FACHIN

12127084

Paul GENEST

12131124

Supervisor:

Dr. Kevin STURM

July 12, 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	6
4	The Lagrangian, State Equation and Geometric Constraints	7
5	Shape Derrivative	8
6	Deformation	9
7	Iteration	10
8	Results and Conclusion	11
	References	12
	Appendices	13
A	Python Code Listing	13
B	XMI Code Listing	14
C	MATLAB Code Listing	15

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

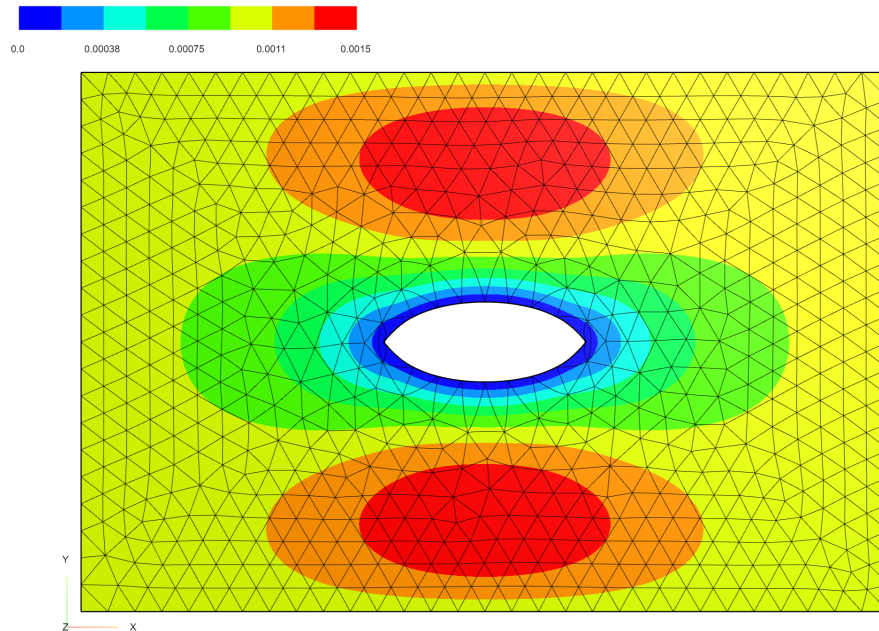
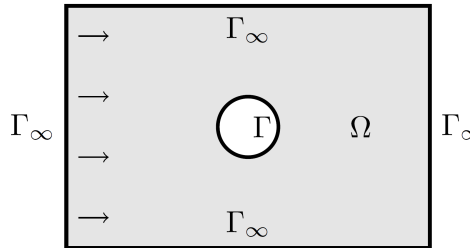


Figure 1: Velocity magnitude of Stokes flow after shape optimization

2 The Stokes Equations in NGSolve

The Stokes Equations are linear partial differential equations, which describe a stationary incompressible Newtonian fluid flow with high viscosities and low Reynolds numbers. For the implementation in NGSolve, a suitable geometry and boundary conditions are the ones proposed by Sturm et. al. [2], where the fluid flow around a cylinder is investigated while the outer boundary of Ω is prescribed a velocity strictly in x direction, the so called far field velocity:



$$\begin{aligned}
 -\mu \Delta u + \nabla p &= 0 & \text{in } \Omega, \\
 \operatorname{div} u &= 0 & \text{in } \Omega, \\
 \mathbf{u} &= 0 & \text{on } \Gamma, \\
 \mathbf{u} &= \mathbf{u}_\infty & \text{on } \Gamma_\infty,
 \end{aligned} \tag{1}$$

Figure 2: Domain Ω for Stokes PDE's (1) [2]

Where $\mu \in \mathbb{R}$ is the viscosity constant and is set to 1 for simplicity. The problem yields the vectorial velocity field $u : \Omega \rightarrow \mathbb{R}^d$ and the scalar pressure field $p : \Omega \rightarrow \mathbb{R}$. 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 derived by multiplying 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

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

Instead of considering this as a system of equations, one can look at the mixed method as one variational problem on the product spaces $[H_0^1(\Omega)]^d \times L^2(\Omega)$, this is done by just adding both problems [5]:

$$\int_{\Omega} \nabla u : \nabla v \, dx + \int_{\Omega} \operatorname{div}(v) p \, dx + \int_{\Omega} \operatorname{div}(u) q \, dx = 0 \quad \forall (v, q) \in [H_0^1(\Omega)]^d \times L^2(\Omega) \tag{3}$$

In lines 19-26 of listing 2, the variational problem 3 is added to a `BilinearForm()`. After assembling of the system, in line 27-31 the non-zero Dirichlet conditions are assigned. When setting up the geometry, the boundaries already have to be named to do the boundary conditions assignment. The geometry shown in figure 2, is defined in the beginning in lines 5-11.

Basic Stokes PDE's with Python3 and NGSolve

```

1
2  from ngsolve import *
3  from netgen.geom2d import SplineGeometry
4  from ngsolve.webgui import Draw
5  # Geometry with meshwidth h_m
6  h_m = 0.4
7  geo = SplineGeometry()
8  geo.AddRectangle((-3,-2), (3, 2), bcs=("top", "out", "bot", "in"), leftdomain=1, rightdomain=0)
9  geo.AddCircle(c=(0, 0), r=0.5, leftdomain=0, rightdomain=1, bc="cyl", maxh=h_m)
10 mesh = Mesh(geo.GenerateMesh(maxh=h_m))
11 mesh.Curve(3);
12 # Setting up appropriate Function Spaces and boundary Conditions
13 k = 2
14 V = H1(mesh,order=k, dirichlet="top|bot|cyl|in|out")
15 Q = H1(mesh,order=k-1)
16 FES = FESpace([V,V,Q]) # Omitting command VectorH1 --> [V,Q]
17 ux,uy,p = FES.TrialFunction()
18 vx,vy,q = FES.TestFunction()
19 # stokes equation
20 def Equation(ux,uy,p,vx,vy,q):
21     div_u = grad(ux)[0]+grad(uy)[1] # custom divergence u
22     div_v = grad(vx)[0]+grad(vy)[1] # custom divergence v
23     return (grad(ux)*grad(vx)+grad(uy)*grad(vy) + div_u*q + div_v*p)* dx
24 a = BilinearForm(FES)
25 a += Equation(ux,uy,p,vx,vy,q)
26 a.Assemble()
27 # Assign non-zero Dirichlet boundary conditions u_inf
28 gfu = GridFunction(FES)
29 uinf = 0.001
30 uinf_c = CoefficientFunction((uinf))
31 gfu.components[0].Set(uinf_c, definedon=mesh.Boundaries("in|top|bot|out"))
32 # Define Linear Equation System
33 def solveStokes():
34     res = gfu.vec.CreateVector()
35     res.data = -a.mat * gfu.vec
36     inv = a.mat.Inverse(FES.FreeDofs())
37     gfu.vec.data += inv * res
38     scene_state.Redraw()
39 # Solve LES and plot norm of u
40 solveStokes()
41 u_vec = CoefficientFunction((gfu.components[0], gfu.components[1]))
42 Draw(u_vec, mesh, "vel", draw_surf=True)

```


Below the solution obtained with NGSolve (see listing 2). On the surface of the cylinder, the no-slip condition (standard Dirichlet = 0) can be observed. Also an intuitive observation of the fulfilled continuity can be made: where the cross section is smaller, e.g. in x vicinity of the cylinder, the velocity is increased:

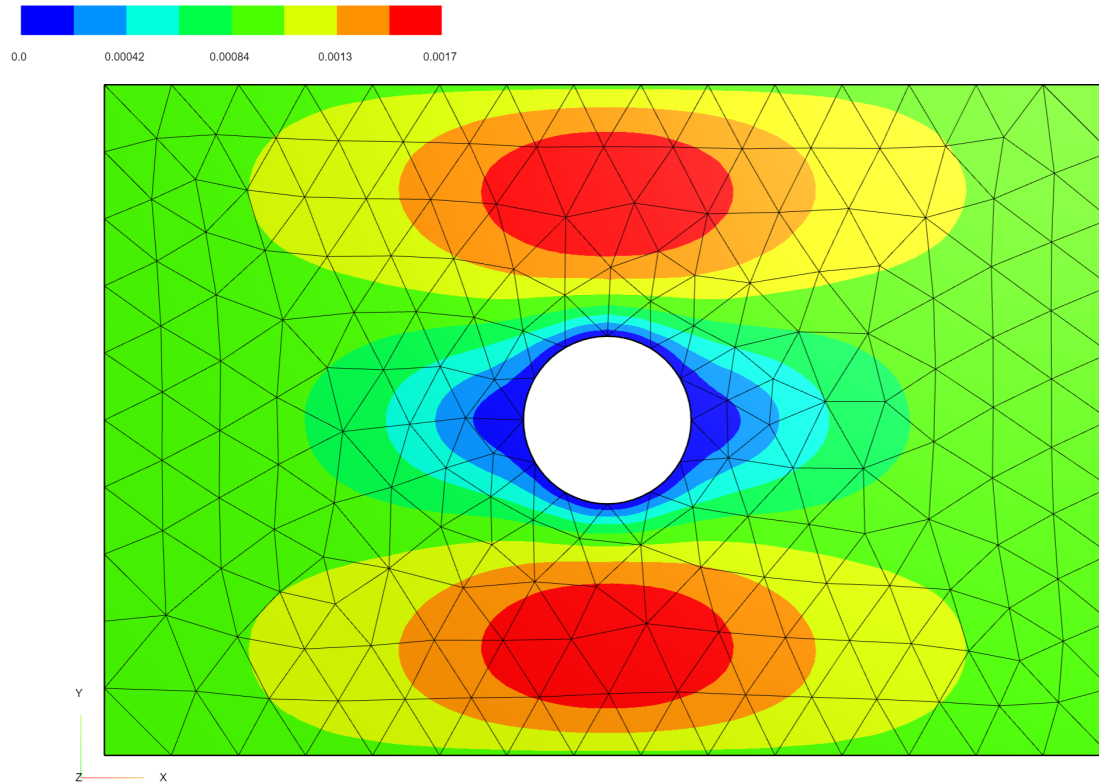


Figure 3: Surface Plot - Velocity $\|\mathbf{u}\|_2$ of Stokes Flow - FEM solution to problem (1)

3 Shape Optimization

Shape optimization is the process of minimizing of shape functions J . This function depends on the domain Ω , which will be perturbed in the minimization process. The perturbations of the shape Ω are described by the following transformation: $\Omega_t := (Id + tX)(\Omega)$. According to this, for small perturbations and $t > 0$, the shape derivative is: [3]

$$DJ(\Omega)(X) := \left(\frac{\partial}{\partial t} J(\Omega_t) \right) \Big|_{t=0} = \lim_{t \rightarrow 0} \frac{J(\Omega_t) - J(\Omega)}{t} \quad (4)$$

This is needed for ... ?

A lot of engineering applications require the shape to be dependent on a PDE. The resulting problems are called PDE constrained shape optimization. This yields a minimization problem of the shape function subjected to the side constraints of the (in our case) stokes equation.

This

4 The Lagrangian, State Equation and Geometric Constraints

Die Terminologie ist dann "state equation" = die Stokes Gleichung. lel

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

```

B XML 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

```
1 %% Linear model Poly44 from MATLAB Curve Fit App:
2
3 %Polynomial Coefficients (with 95\% confidence bounds):
4     p00 =      13.79;  %(13.22, 14.36)
5     p10 =     -2.897; %(-3.454, -2.34)
6     p01 =      3.752; %(3.163, 4.34)
7     p20 =      3.279; %(2.231, 4.327)
8     p11 =      0.5404; %(-0.2001, 1.281)
9     p02 =      0.8638; %(-0.4624, 2.19)
10    p30 =      0.299;  %(0.01281, 0.5851)
11    p21 =     -0.5091; %(-0.7299, -0.2884)
12    p12 =      0.4973; %(0.2716, 0.7229)
13    p03 =      0.3595; %(0.04484, 0.6741)
14    p40 =     -0.8495; %(-1.291, -0.4084)
15    p31 =     -0.02258; %(-0.3136, 0.2685)
16    p22 =     -0.2819; %(-0.5502, -0.01351)
17    p13 =      0.2674; %(-0.05265, 0.5874)
18    p04 =      0.2019; %(-0.3968, 0.8006)
19
20    f(x,y) = p00 + p10*x + p01*y + p20*x^2 + p11*x*y + p02*y^2 + p30*x^3 + p21*x^2*y
21    + p12*x*y^2 + p03*y^3 + p40*x^4 + p31*x^3*y + p22*x^2*y^2
22    + p13*x*y^3 + p04*y^4
23
24    %Goodness of fit:
25    %SSE: 3.189
26    %R-square: 0.9949
27    %Adjusted R-square: 0.9902
28    %RMSE: 0.4611
```