

Chapter 5

Numerical Optimization in Microfluidics

Kristian Ejlebjerg Jensen

Abstract Numerical modelling can illuminate the working mechanism and limitations of microfluidic devices. Such insights are useful in their own right, but one can take advantage of numerical modelling in a systematic way using numerical optimization. In this chapter we will discuss when and how numerical optimization is best used.

5.1 Introduction

Within complex fluid-flow, the use of numerical optimization is still a rarity, and thus it is only the most obvious applications, such as [2, 9] rectifiers and cross-slot geometries [7, 10], that have been the subject of numerical optimization.

Numerical modelling of complex fluids is often a challenging task in itself. This goes for viscoelastic fluids in particular, but even generalized-Newtonian models can cause numerical instabilities. Numerical optimization builds on top of modelling, so in order for it to be successful, the model has to be robust towards large variations in the design. I.e. the numerical model is your foundation, so a critical prerequisite for performing numerical optimization is that you

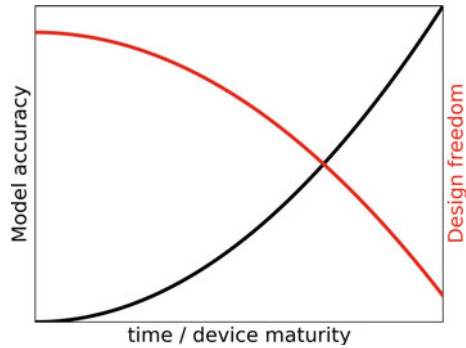
verify the robustness of your numerical model and never perform optimization outside its limits—the optimization result should be independent of the numerical discretization.

Ideally, you should also be able to rate the accuracy of your mathematical model, but you might be able to benefit from an optimization based on an inaccurate model, if you have a lot of design freedom. This is due to the fact that such an optimization has the potential to identify new working mechanisms for your design. Conversely, it is

K.E. Jensen (✉)

Department of Micro- and Nanotechnology, Technical University of Denmark,
Ørsteds Plads, 2800 Kongens Lyngby, Denmark
e-mail: krej@dtu.dk

Fig. 5.1 Large design freedom and low model accuracy typically characterizes the early phase of device development, while the opposite is true for later stages. One can thus benefit from numerical optimization throughout the development process, but the potential gain decreases with design freedom



also true that you can benefit from an optimization with little design freedom, if the model is very accurate. The potential gain will be smaller and likely of an evolutionary character, but the probability that the optimization will actually be implemented is high. Note, however, that

you are unlikely to benefit from an optimization, if you have an inaccurate model and little design freedom.

Such an optimization will result in a small perturbation to the existing design, so it will not identify a new working mechanism and the optimality of the perturbation cannot be trusted due to the inaccuracy of the model. Although early optimization is preferable, it is also worthwhile to note that the understanding of parameters and flow effects increase as a device is developed, which improves the accuracy of the model, see Fig. 5.1.

5.2 Variables

An optimization problem consists of a model, variables, constraints and an objective function. The objective function is used to rate the design and by convention it should be minimized. This is achieved by changing the variables, while respecting the constraints (and the model).

The variables often describe the design, but in principle they can be anything, so the viscosity or other material parameters can also be allowed to vary. Ultimately the choice of variables depends on what you are able to realize, so this choice is strongly tied to the degree of design freedom, whether it be manufacturing constraints or the parameter regime within which the model validity has been verified.

Most optimizers work best, when there are no constraints associated with the variables, which is rarely possible. The second best option is to have a formulation with lower and upper bounds (*box constraints*), so

try to select the design variables of your problem such that box constraints can be used.

To achieve this, one might have to reformulate the variables. In example one could consider the flow past a sphere in a tube with radii r and R , respectively. If the sphere is to fit in the tube, one might impose

$$0 < R < R_{\max} \quad \text{and} \quad 0 < r < R$$

as constraints, but the last inequality is not a box constraint, so it is better not to use r as a design variable. Instead one can have a design variables a such that

$$r = aR, \quad \text{where} \quad 0 < a < 1$$

This way box constraints can be used to ensure that the sphere fits in the tube.

5.3 Simultaneous Analysis and Design (SAND)

There are two types of optimization techniques:

- #1 The nested formulation, where one alternates between computing the physical variables for the current design and updating the design variables based on the current result of the numerical model.
- #2 Simultaneous analysis and design (SAND), where there is no distinction between the design variables and the physical variables. This means that the governing equations are treated as constraints.

The SAND approach should in theory be able to converge much faster and this has also been demonstrated for problems within fluid dynamics [5], but the method cannot guarantee improvement on an initial design and often fails to even satisfy the governing equation for all, but the most simple problems. It is thus more of an interesting research topic than a practical tool for applied optimization problems. In the following, we will thus restrict ourselves to the nested formulation.

5.3.1 Non-parametric Optimization

The number of variables does not have to be tied to a fixed set of design features. It can be formulated in a more abstract sense, such that the boundary is allowed to vary by having it defined implicitly as the contour of a spatially varying field, i.e. a *level-set* function. Alternatively, one can use an explicit boundary representation,

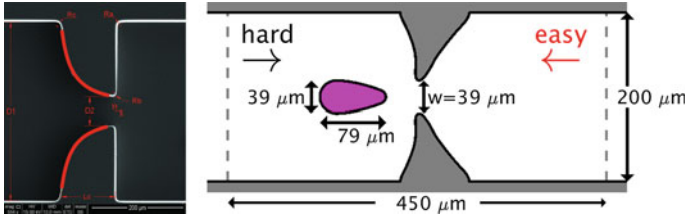


Fig. 5.2 The figure to the *left* shows a typical contraction, which has been extensively studied for its direction dependent hydraulic resistance. The design to the *right* has been studied in the same context, but it features a qualitatively different design, which is the result of topology optimization. The figures are reproduced from [4, 8]

but this normally prevents changes of the design topology. In any case, the number of variables for such an optimization will scale with the numerical discretization of the underlying model, which implies thousands of design variables and thus also a restricted set of applicable optimization methods. The advantage of these methods is that they allow for extreme design freedom such that the qualitative layout of the design (i.e. the number of holes / the design topology) does not have to be known a priori, hence the name *topology optimization*, see Fig. 5.2.

Examples of this include micro reactors [11] as well as inertial- and viscoelastic rectifiers [8, 12]. If one finds a design with a novel topology using non-parametric optimization, it is a good idea to perform a parametric optimization as a post-processing step. This can simplify fabrication and enable other researchers to reproduce the experiments. Finally, it is always advisable to

understand the working mechanism of the optimal design and reconsider your problem statement with this in mind.

5.4 Objective Function and Constraints

The choice of objective function, constraints and problem formulation are intimately related. In example, many devices will benefit from the energy that is put into them, so if one imposes fixed in-flow boundary conditions, there is a possibility that the optimization will result in a design with extremely high hydraulic resistance, effectively blocking the system. One can get around this by imposing a pressure drop constraint, or simply by switching to boundary conditions with a fixed pressure drop.

Micro devices for complex fluids tend to rely on effects that become (relatively) stronger at small length scales. This, however, also means that an optimization might try to introduce small length scales and one thus have to consider the choice of design variables and their constraints carefully. In a non-parametric optimization, one might

have to introduce a minimum curvature or minimum length scale. In any case, one has to prevent the optimizer from making structures smaller than what can be both accurately captured by the model and experimentally realized.

Finding the best problem statement might involve some trial and error, but

when you identify potential objective functions and constraints of your problem, you should keep in mind that trivial designs are never optimal for well-posed problem formulations.

I.e. if you want to minimize the viscous dissipation in a channel, the optimal design will either be a complete open or a completely blocked channel depending on whether the flow rate or the pressure drop is fixed, so unless you introduce a volume constraint, you will get a trivial design.

For inequality constraint functions the convention is that they should be negative and the set of design variables respecting all constraints is called the feasible set. Some optimization algorithms satisfy inequality constraints, g_i , by minimizing a modified objective function

$$O' = O + \sum_i^N w_i g_i,$$

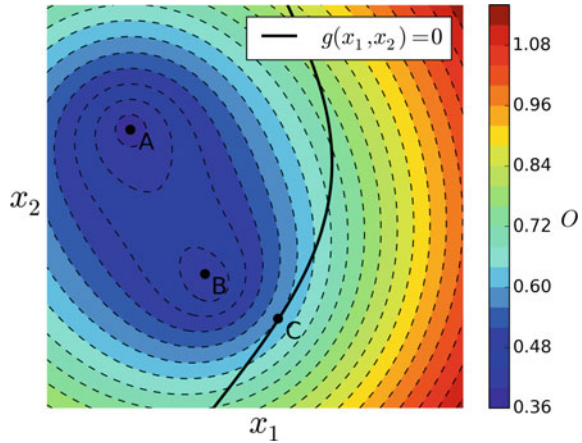
where w_i are weights, which are kept as small possible and only increased, when a constraint is violated. The details of such a procedures often involve some heuristics and assumptions about functions and variables. This means that such

general purpose mathematical optimizers work best, if the problem is stated in a non-dimensional formulation.

Furthermore, equality constraints are often treated as two inequality constraints in the numerical implementation and they always make the problem stiff, so that the maximum step size of the optimizer is severely reduced.

It is a good idea to investigate the smoothness of the problem and whether the objective function is convex or not. If the problem has several minima, as illustrated in Fig. 5.3, it is non-convex and if there are many local minima, it will be difficult to find a good design. Within non-parametric optimization it is common to solve a convex problem, which is similar, but not identical, to the actual problem one wants to solve. One can then make a continuation from the easy and wrong problem to the hard and correct, so that a good design can be found. This tip, however, comes with a footnote:

Fig. 5.3 An objective function O is drawn in a contour plot as a function of two design variables, x_1 and x_2 . The $g(x_1, x_2) = 0$ contour of the constraint function is also sketched and depending on whether the feasible region ($g < 0$) is on the left or right side of the plot, the optimal design is either A or C, respectively. B is a local extremum, so this is an example of a non-convex optimization problem



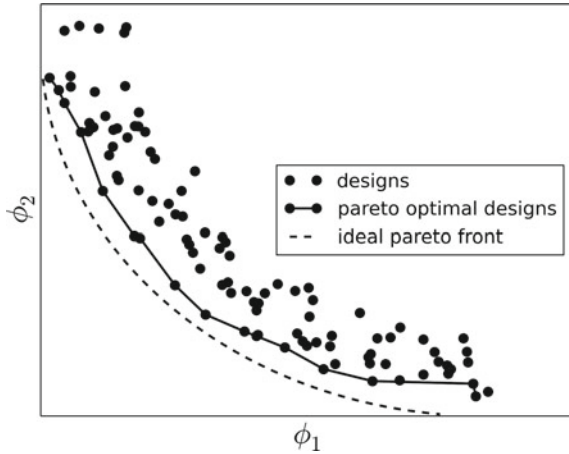
Local minima are best avoided using continuation methods and multiple initial designs guesses. They, however, tend to be more robust towards parameter- or design variations.

If one wants to take robustness into account in a more systematic way, a range of physical parameters can be considered or different perturbations to the *blueprint design* can be investigated. If several such models are used to construct an objective function, the robustness of the global minimum is likely to improve. The actual value, however, will increase—lunch is never free.

5.4.1 Pareto-Optimality

Sometimes there are several critical objective functions, and it is impossible to choose how to prioritise them, before the optimization has been carried out. This is because one wants to know how much can be gained of one objective by giving up a certain amount of another objective, i.e. you accept that *there is no free lunch*, but you want to know the cost. In such a case one will typically resort to multi objective optimization. This involves an objective function based on a weighted average and a detailed study of the effects of the weights. Plotting the objectives as functions of each other will then reveal the pareto optimal front as shown in Fig. 5.4. This also indicates whether some of the optimizations resulted in local minima.

Fig. 5.4 In the context of multiobjective optimization, a design is said to be pareto optimal, when no other design is better in terms of all objectives. The pareto front consists of such designs as illustrated for a problem with two objective functions, ϕ_1 and ϕ_2



5.5 Gradient Free Methods

The most simple optimization one can imagine involves mapping out the entire design space. This is a robust approach that is easy to implement, but it is also very expensive in terms of computational resources. This downside can be somewhat mitigated by using an initial coarse map to select a subregion for further analysis in what can end up as a hierarchical method, but ultimately such an approach is unlikely to be attractive, if one has more than a handful of variables.

Powell's method is a simple gradient free method, which works by

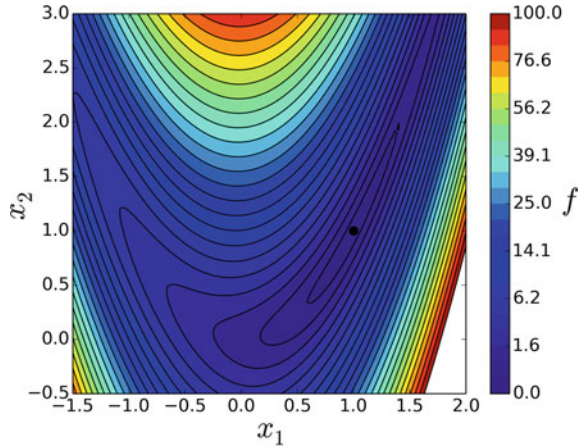
- #1 Pick a starting guess \mathbf{x}_0 and a number of search vectors \mathbf{v}_i .
- #2 Perform a line search along each search vector, i.e. find the minimum of $O(w_i \mathbf{v}_i + \mathbf{x}_0)$ with respect to w_i .
- #3 Update the guess to $\mathbf{x}_0 + \sum w_i \mathbf{v}_i$ and replace the search vector having the lowest $|w_i|$ with $\sum_i w_i \mathbf{v}_i$. Go to #2.

The search vector substitution is critical for the performance in problems with strong anisotropy, such as the Rosenbrock function, see Fig. 5.5. The complexity of the method lies in the line search, which is also the only part of the algorithm involving function evaluations. The line searches can be performed independently of each other, which makes it trivial to realize a parallel implementation of the algorithm.

The downhill simplex method or Nelder-Mead method is an alternative, which is difficult to realise in a parallel implementation. The concept is also somewhat more complicated, but there is no linear search tolerance to be set and perhaps updated, which is probably why it is the most popular gradient free method. It works using a simplex that is reflected, expanded, contracted and shrunk so as to find the minimum of the objective function.¹ The only parameters of the method are related to the

¹Animations available at https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method. Cited 30 April 2017.

Fig. 5.5 The Rosenbrock function $f(x_1, x_2) = (1 - x_1)^2 + 10(x_2 - x_1^2)^2$ has a minimum at $x_1 = x_2 = 1$, but the anisotropic nature of the function makes it difficult to locate this minimum, and therefore the function it is a popular benchmark problem for optimization methods



stopping criteria and it is one of the method without internal parameters. Internal parameters can be problematic for poorly scaled problems, so the simplex method is thus an exception to the rule that general purpose mathematical optimizers work best, if the problem is stated in a non-dimensional formulation.

Both the Powell's and the Nelder-Mead method is available through python's `scipy.optimize.minimize` function. Any constraints will have to be enforced by setting the objective function to infinity outside the feasible region.

It is important to test whether different starting guesses results in different minima. The global minimum of the numerical objective function will also change whenever the topology of the discretization is changed, but this effect is reduced with a finer discretization as shown to the left in Fig. 5.6. For a mesh based model this means that the objective function is only smooth, if the connectivity is fixed, but the vertices are allowed to move. Such a strategy has been used to optimize a viscoelastic rectifier [2]. If a minimum is far from the starting guess, the discretization will have to morph a lot, which can be difficult to achieve and reduce the accuracy of the model. In that situation, one should consider making a new discretization and restarting the optimization. To ease this process you should

consider accelerating your parametric studies by using software featuring automatic spatial discretization/meshing.

When using a gradient free method, one can in principle do this in every iteration as illustrated to the right in Fig. 5.6.

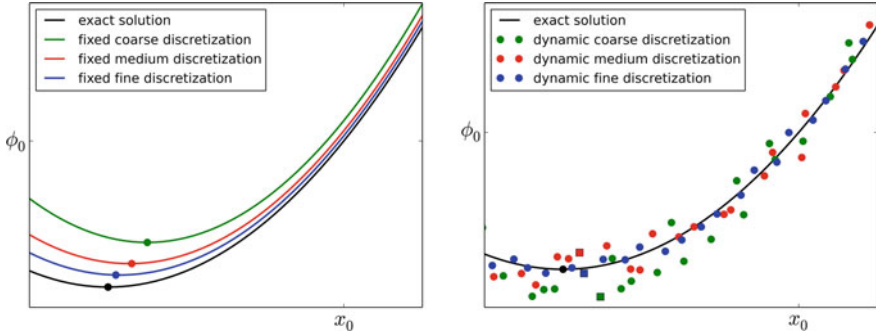


Fig. 5.6 An objective function ϕ is plotted as a function of a variable x . To the *left* a discretization optimized for $x = x_0$ is used, which leads to a low deviation from the exact solution at this point. To the *right* the deviation is low everywhere, because the discretization is changed for every x , but this means that one cannot estimate the gradient accurately using finite differences

5.6 Gradient Based Methods

Gradient based methods require that you can construct a smooth representation of the objective function (at least locally), such that

$$\phi(\mathbf{x}) = \phi(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0) \cdot \nabla\phi|_{\mathbf{x}=\mathbf{x}_0} + \mathcal{O}((\mathbf{x} - \mathbf{x}_0)^2)$$

i.e. a 1st order Taylor expansion. The most simple way to calculate the components of the gradient is using finite differences, i.e.

$$\begin{aligned} (\nabla\phi|_{\mathbf{x}=\mathbf{x}_0})_i &= \frac{\phi(\mathbf{x}_0 + \delta\mathbf{x}_i) - \phi(\mathbf{x}_0)}{\delta x} + \mathcal{O}(\delta x), && \text{forward difference} \\ &= \frac{\phi(\mathbf{x}_0 + \delta\mathbf{x}_i) - \phi(\mathbf{x}_0 - \delta\mathbf{x}_i)}{2\delta x} + \mathcal{O}(\delta x)^2, && \text{central difference,} \end{aligned}$$

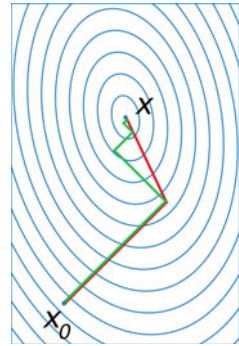
where $\delta\mathbf{x}_i$ is a small variation in the i 'th variable.² One can use this technique to

test the accuracy of sensitivities, if they are computed without using finite differences.

The most simple approach to gradient based optimization is gradient descent, which boils down to a line search along the gradient direction. More interesting is the conjugate gradient (CG) method, which is guaranteed to converge in the same

²A value three orders of magnitude larger than the machine precision is a good starting point, but the optimal value is problem dependent and it is thus a good idea to study, when numerical noise dies out and 2nd order effects sets in.

Fig. 5.7 A gradient descent algorithm is compared to the conjugate gradient method for a quadratic function in 2D. It will thus converge in 2 iterations, once it is close to a minimum. https://en.wikipedia.org/wiki/Conjugate_gradient_method. Cited 30 April 2017



number of iterations as the number of design variables, if the objective function is quadratic (which is generally the case close to the minimum), see Fig. 5.7. The CG method is also available through python's `scipy.optimize.minimize` function.

If one can also calculate the Hessian, the 2nd order Taylor expansion can be constructed. It is straightforward to compute the minimum of this quadratic function and iterating in this way is called Newton's method. This has quadratic order asymptotic convergence, which means that the logarithm of the error is halved in every iteration, when the method is close to an optimum. In comparison, gradient descent will only halve the actual error in every iteration.

For most applications, the convergence rate of a method is irrelevant, since this only applies close to the minimum, while most of the computational time tends to be spent far from the minimum. If faced with the question of which optimization algorithm to pick for some arbitrary problem, it is thus relevant to look at the performance for a wide range of benchmark problems as those found in the CUTer (Constrained and Unconstrained Testing Environment, revisited) set, which contains around 1,000 optimization problems. The Ipopt package does well on this test set [3], it is open source and available through the PyIpopt python module.³ The method of moving asymptotes is an alternative that is very popular within structural optimization [13], but it is only free for academic use.

5.6.1 Non-parametric Optimization

Non-parametric optimization methods tend to involve many design variables and therefore one has to use efficient techniques for the computation of the gradient. The adjoint variable technique is the only method for doing this, either in a discrete or a continuous version. Both can give the exact gradient for a given discretization, but it

³<https://github.com/xuy/pyipopt>. Cited 30 April 2017.

is only guaranteed for the discrete versions. Dolfin adjoint is a module for the FEniCS [1] finite element package, and it can calculate the discrete gradient automatically [6]. In general,

Software capable of automatic differentiation is a great help for non-linear problems as well as for computation of gradients.

The advantage of the continuous adjoint technique is that it is independent of numerical methods and thus potentially provides insight on a higher level. As an example, we will consider Stokes flow with a Brinkman damping term. The sensitivity of the viscous dissipation with respect to local changes in damping coefficient can be calculated using the continuous adjoint technique as follows. The governing equations and objective function, ϕ are

$$\phi = \int_{\Omega} \left(\frac{1}{2} \underline{\underline{\varepsilon}} : \underline{\underline{\sigma}} - \mathbf{u} \cdot \mathbf{F} \right) d\Omega \quad (5.1)$$

$$\mathbf{0} = \nabla \cdot \underline{\underline{\sigma}} + \mathbf{F} \quad (5.2)$$

$$0 = \nabla \cdot \mathbf{u} \quad \text{where} \quad (5.3)$$

$$\underline{\underline{\sigma}} = -\mathbf{I}p + \eta \underline{\underline{\varepsilon}}, \quad \underline{\underline{\varepsilon}} = \nabla \mathbf{u} + [\nabla \mathbf{u}]^T$$

$$\mathbf{F} = -\alpha \mathbf{u},$$

where \mathbf{u} , p , η , \mathbf{F} , $\underline{\underline{\sigma}}$, $\underline{\underline{\varepsilon}}$ and α are the velocity, pressure, viscosity, volumetric force, stress, rate of deformation and damping term, respectively. It is easy to see that the objective function can be simplified due to Eq. (5.3),

$$\phi = \int_{\Omega} \left(\frac{1}{2} \eta \underline{\underline{\varepsilon}} : \underline{\underline{\varepsilon}} - \mathbf{u} \cdot \mathbf{F} \right) d\Omega, \quad \text{because} \quad \underline{\underline{\varepsilon}} : \mathbf{I} = 2\nabla \cdot \mathbf{u} = 0 \quad (5.4)$$

A variation in the damping term, $\delta\alpha$ will result in a variation of the objective function,

$$\begin{aligned} \delta\phi &= \int_{\Omega} \left(\eta \frac{\partial \underline{\underline{\varepsilon}}}{\partial \alpha} : \underline{\underline{\varepsilon}} - \frac{\partial \mathbf{u}}{\partial \alpha} \cdot \mathbf{F} - \mathbf{u} \cdot \frac{\partial \mathbf{F}}{\partial \alpha} \right) \delta\alpha d\Omega \\ &= \int_{\Omega} \left(\eta \frac{\partial \underline{\underline{\varepsilon}}}{\partial \alpha} : \underline{\underline{\varepsilon}} + 2\alpha \mathbf{u} \cdot \frac{\partial \mathbf{u}}{\partial \alpha} + \mathbf{u}^2 \right) \delta\alpha d\Omega \end{aligned} \quad (5.5)$$

This can be expanded further, but the point is that the derivatives of \mathbf{u} and p with respect to α are unknown and therefore the derivative of $\underline{\underline{\varepsilon}}$ is also unknown, so they somehow have to be eliminated. This can be achieved by constructing other partial differential equations with the same terms and adding/subtraction the equations. The

starting point of this procedure is the introduction the adjoint variables $\tilde{\mathbf{u}}$ and \tilde{p} , which by definition are invariant with respect to the $\delta\alpha$ variation. Multiplying $\tilde{\mathbf{u}}$ with Eq. (5.2) and integrating over the domain, Ω , yields

$$\begin{aligned} 0 &= \int_{\Omega} \tilde{\mathbf{u}} \cdot (\nabla \cdot \underline{\underline{\sigma}} + \mathbf{F}) d\Omega \\ &= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \underline{\underline{\sigma}} \cdot \hat{\mathbf{n}} ds - \int_{\Omega} (\nabla \tilde{\mathbf{u}} : \underline{\underline{\sigma}} + \tilde{\mathbf{u}} \cdot \mathbf{F}) d\Omega \\ &= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \underline{\underline{\sigma}} \cdot \hat{\mathbf{n}} ds - \int_{\Omega} \left(\frac{1}{2} \tilde{\underline{\underline{\varepsilon}}} : \underline{\underline{\sigma}} - \tilde{\mathbf{u}} \cdot \mathbf{F} \right) d\Omega \quad \text{where} \quad \tilde{\underline{\underline{\varepsilon}}} = \nabla \tilde{\mathbf{u}} + [\nabla \tilde{\mathbf{u}}]^T \end{aligned} \quad (5.6)$$

where we have used the divergence theorem and the fact that the stress tensor is symmetric. Taking the variation with respect to $\delta\alpha$ yields

$$\begin{aligned} 0 &= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \frac{\partial \underline{\underline{\sigma}}}{\partial \alpha} \cdot \hat{\mathbf{n}} \delta\alpha ds - \int_{\Omega} \left(\frac{1}{2} \tilde{\underline{\underline{\varepsilon}}} : \frac{\partial \underline{\underline{\sigma}}}{\partial \alpha} - \tilde{\mathbf{u}} \cdot \frac{\partial \mathbf{F}}{\partial \alpha} \right) \delta\alpha d\Omega \\ &= \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \frac{\partial \underline{\underline{\sigma}}}{\partial \alpha} \cdot \hat{\mathbf{n}} \delta\alpha ds \\ &\quad - \int_{\Omega} \left(\frac{1}{2} \tilde{\underline{\underline{\varepsilon}}} : \left[-\mathbf{I} \frac{\partial p}{\partial \alpha} + \eta \frac{\partial \underline{\underline{\varepsilon}}}{\partial \alpha} \right] + \tilde{\mathbf{u}} \cdot \left[\alpha \frac{\partial \mathbf{u}}{\partial \alpha} + \mathbf{u} \right] \right) \delta\alpha d\Omega \end{aligned} \quad (5.7)$$

It is now easy to see that we can eliminate all derivatives with respect to α by adding Eqs. (5.5) and (5.7) and assuming $\tilde{\mathbf{u}} = 2\mathbf{u}$. This means that $\tilde{\underline{\underline{\varepsilon}}} : \underline{\underline{\mathbf{I}}} = 2\nabla \cdot \tilde{\mathbf{u}} = 0$, so we get

$$\delta\phi = \int_{\partial\Omega} \tilde{\mathbf{u}} \cdot \frac{\partial \underline{\underline{\sigma}}}{\partial \alpha} \cdot \hat{\mathbf{n}} \delta\alpha ds + \int_{\Omega} -\mathbf{u}^2 \delta\alpha d\Omega$$

The boundary term drops out, if we restrict ourselves to no-slip boundary conditions and inlet/outlets with fixed pressure and zero normal viscous stress. This is due to the fact that either $\tilde{\mathbf{u}} = 2\mathbf{u} = \mathbf{0}$ or $\underline{\underline{\sigma}} \cdot \hat{\mathbf{n}} = \hat{\mathbf{n}} p_{\text{bnd}}$. The sensitivity of the objective function thus becomes

$$\frac{\partial \phi}{\partial \alpha} = -\mathbf{u}^2$$

The fact that the adjoint velocity can be expressed explicitly from the physical velocity means that the problem is *self-adjoint*, which is a special case. In general the sensitivity will depend on the physical as well as the adjoint variables with separate partial differential equations and associated boundary conditions for the adjoint variables. The adjoint equations are, however, guaranteed to be linear. For an example of this refer to the Appedix A.

5.7 Summary

In this chapter we have covered when optimization makes sense and how it is best carried out. The following key tips have been given

- #1 Verify the robustness of your numerical model and never perform optimization outside its limits—the optimization result should be independent of the numerical discretization.
- #2 You are unlikely to benefit from an optimization, if you have an inaccurate model and little design freedom.
- #3 Try to select the design variables of your problem such that box constraints can be used.
- #4 Understand the working mechanism of the optimal design and reconsider your problem statement with this in mind.
- #5 When you identify potential objective functions and constraints of your problem, you should keep in mind that trivial designs are never optimal for well-posed problem formulations.
- #6 Local minima are best avoided using continuation methods and multiple initial designs guesses. They, however, tend to be more robust towards parameter- or design variations.
- #7 General purpose mathematical optimizers work best, if the problem is stated in a non-dimensional formulation.
- #8 Consider accelerating your parametric studies by using software featuring automatic spatial discretization/meshing.
- #9 Test the accuracy of sensitivities, if they are computed without using finite differences.
- #10 Software capable of automatic differentiation is a great help for non-linear problems as well as for computation of gradients.

Other points relate to multi objective optimization, the nested optimization formulation and the adjoint technique for computing sensitivities.

5.8 Exercises

Discussion exercises

Answer the following questions with regards to a project

- #1 Do you have a model? If not, do you know enough about your system to construct a model? How accurate and robust is or could it be? Do you have a lot of design freedom?
- #2 Discuss possible objective functions, variables and constraints for your project.
- #3 Do you understand the working mechanism of your system? Can you predict the outcome of an optimization in a qualitative sense?

Numerical exercises

Perform the following exercises in your software of choice

- #1 Implement Powel's method and test it for the Rosenbrock function. Plot the path for different initial guesses.
- #2 Reuse one of your models from the previous chapter and choose an objective function and a constraint. Plot the relation between the two with a fixed and dynamic mesh topology.
- #3 Use an optimization method to find the minimum automatically.

Acknowledgements This work is supported by the Villum Foundation (Grant No. 9301) and the Danish Council for Independent Research (DNRF122).

References

1. Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., et al. (2015). The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100): 9–23.
2. Alves, M.A. (2011). Design of optimized microfluidic devices for viscoelastic fluid flow. In *Technical Proceedings of the 2011 NSTI Nanotechnology Conference and Expo. 2*: 474–477.
3. Bongartz, I., Conn, A. R., Gould, N., & Toint, Ph L. (1995). CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software (TOMS)*, 21(1), 123–160.
4. Campo-Deaño, L., Galindo-Rosales, F. J., Pinho, F. T., Alves, M. A., & Oliveira, M. S. N. (2011). Flow of low viscosity Boger fluids through a microfluidic hyperbolic contraction. *Journal of Non-Newtonian Fluid Mechanics*, 166(21), 1286–1296.
5. Evgrafov, A. (2015). On Chebyshev's method for topology optimization of Stokes flows. *Structural and Multidisciplinary Optimization*, 51(4), 801–811.
6. Farrell, P. E., Ham, D. A., Funke, S. W., & Rognes, M. E. (2013). Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4), C369–C393.
7. Galindo-Rosales, F.J., Oliveira, M. S. N., & Alves, M. A. (2014). Optimized cross-slot microdevices for homogeneous extension. *RSC Advances*, 4(15), 7799–7804.
8. Jensen, K. E., Szabo, P., Okkels, F., & Alves, M. A. (2012). Experimental characterisation of a novel viscoelastic rectifier design. *Biomicrofluidics*, 6(4), 044112.
9. Jensen, K. E., Szabo, P., & Okkels, F. (2012). Topology optimization of viscoelastic rectifiers. *Applied Physics Letters*, 100(23), 234102.
10. Jensen, K. E., Szabo, P., & Okkels, F. (2014). Optimization of bistable viscoelastic systems. *Structural and Multidisciplinary Optimization*, 49(5), 733–742.
11. Krühne, U., Larsson, H., Heintz, S., Ringborg, R. H., Rosinha, I. P., Bodla, V. K., et al. (2014). Systematic development of miniaturized (bio) processes using Process Systems Engineering (PSE) methods and tools. *Chemical and Biochemical Engineering Quarterly*, 28(2), 203–214.
12. Lin, S., Zhao, L., Guest, J. K., Weihs, T. P., & Liu, Z. (2015). Topology optimization of fixed-geometry fluid diodes. *Journal of Mechanical Design*, 137(8), 081402.
13. Svanberg, K. (1987). The method of moving asymptotes-A new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2), 359–373.