Chapter 4

Single-Grid Optimization

4.1 Introduction

In this section, we recall some well-known unconstrained optimization methods. Our purpose is to emphasize some aspects of the implementation and application of these methods in the case of PDE optimization. In fact, apart from being large-sized, PDE optimization problems are naturally formulated on functional spaces where the objectives and the solution of the PDE model are defined. Thus, depending on the inner product and norms used, the formulation and implementation of any optimization method will change. For an excellent introduction to optimization methods in Euclidean spaces see [265].

Here, black-box methods represent the class of optimization schemes that apply to any optimization problem independently of the nature and structure of the problem, provided that some essential operators are available. Specifically, we assume that corresponding to a given value of the optimization variable, we can evaluate the objective and its gradient with respect to this variable. In addition, we may require computing the product of the Hessian with a given vector. In PDE optimization, these operations require a particular effort.

4.2 Black-Box Methods

Consider the following class of PDE optimization problems

$$\left\{ \begin{array}{rcl} \min J(y,u) & := & h(y) + \nu\, g(u), \\ c(y,u) & = & 0, \end{array} \right. \tag{4.1}$$

where $c : Y \times U \to Z$ represents the PDE operator and the objective $J : Y \times U \to \mathbb{R}$, with appropriate Hilbert spaces Y, U, and Z. We denote with $y \in Y$ the state variable, and $u \in U$ denotes the optimization variable. The solution of the PDE equation provides the differentiable mapping $u \to y(u)$. Here, g and h are required to be continuously differentiable, bounded from below, and such that $g(u) \to \infty$ as $\|u\| \to \infty$. Further, denote by $\hat{J}(u) = J(y(u),u)$ the reduced objective and assume that $u \to \hat{J}(u)$ is uniformly convex. This implies existence of a unique optimizer. The corresponding optimal solution is characterized as the solution to the first-order necessary conditions given by the following optimality system

$$\begin{aligned} c(y,u) &= 0, \\ c_y(y,u)^* p &= -h'(y), \\ \nu\, g'(u) + c_u^* p &= 0. \end{aligned} \tag{4.2}$$

Notice that the third equation of this system represents the optimality condition, in the sense that

$$\hat{J}'(u^*)\delta u = (\nabla \hat{J}(u), \delta u)_U := (\nu\, g'(u) + c_u^* p, \delta u)_U = 0$$

for all variations δu.

We see that the solution of the PDE problem given by the mapping $u \to y(u)$ allows us to transform the PDE-constrained optimization problem in an unconstrained optimization problem as follows

$$\min_{u \in U} \hat{J}(u). \tag{4.3}$$

Thus the PDE equation has formally disappeared, although we need to solve the PDE problem to evaluate the objective at u. Moreover, to compute $\nabla \hat{J}(u)$ for a given u, we have to first solve the state equation and then the adjoint equation to obtain the adjoint variable to construct the gradient. This procedure is summarized in the following.

ALGORITHM 4.1. Evaluation of the gradient at *u*.

```
1. Solve (exactly) the state equation $c(y,u) = 0$.
2. Solve (exactly) the adjoint equation $c_y(y,u)^* p = -h'(y)$;
3. Compute the gradient $\nabla \hat{J}(u) = \nu\, g'(u) + c_u^* p$;
4. End.
```

Now, we focus on black-box numerical methodologies that apply to (4.3) assuming that we solve exactly all related PDE equations approximated by a discretization scheme. Clearly, under these working conditions the optimization properties of the black-box schemes do not depend on the size and structure of the PDE equations except that this structure determines the properties of the underlying Hessian. On the other hand, it is not reasonable to require the exact solution of PDE problems, and this fact represents the true limitation in the use of black-box methods.

Most of these methods implement as the first iteration step a step in the direction of the negative gradient. Recalling the discussion in Section 2.2 about the dependency of the gradient on the metric chosen, one has to decide about a proper scalar product. Of course, the Hessian would be an ideal choice for the scalar product, and in a function space setting, one often has good idea about it, as we see, e.g., in Sections 2.2 and 7.2.1. In finite dimensions, often one does not have the faintest idea about the Hessian, which is then an appropriate positive definite matrix A, if we, e.g., define the scalar product in $\mathbb{R}^2$ as $(u,v)_A = u^\top A v$. This scalar product is equivalent to a change of variables (i.e., scaling) in the form

$$\bar{u} = A^{1/2} u$$

such that the standard scalar product in $\bar{u}$ is equal to the scalar product in u defined by A. It has proved successful in many applications to heuristically choose the scaling in such a manner that the first gradient does not change the control by more than 10%. That means $A^{1/2}u = (s_1u_1,\ldots,s_nu_n)^\top, s_i \in \mathbb{R}_+$, so that $\frac{1}{s_i^2}|\frac{\partial f}{\partial u_i}| \leq 0.1\cdot|u_i|$ and therefore $s_i = \sqrt{10\cdot|\frac{\partial f}{\partial u_i}/u_i|}$ if the expression under the square root is nonzero (otherwise $s_i = |u_i|$) and $|u_i| \neq 0$ (otherwise $s_i = 1$). This scaling strategy has proved invaluable in practical applications like [143, 312].

In the following, we discuss the steepest descent, nonlinear conjugate gradient (NCG), quasi-Newton, and Newton methods and should keep in mind that these methods possess increasingly local convergence properties from linear over superlinear to quadratic [265].

4.2.1 Steepest Descent and NCG Methods

The steepest descent method represents the simplest gradient-based optimization scheme and surely the most popular scheme that is used to validate an optimization framework. The steepest descent algorithm is given in Algorithm 4.2.

ALGORITHM 4.2. Steepest descent scheme.

```
• Input: initial approx. $u_0$, $d_0 = -\nabla\hat{J}(u_0)$, index $k = 0$, maximum $k_{max}$, tolerance $tol$.
    1. While ($k < k_{max}$ && $\|d_k\|_U > tol$ ) do
    2. Evaluate steepest descent $d_k = -\nabla\hat{J}(u_k)$;
    3. Compute steplength $\alpha_k$ along $d_k$ by a given rule;
    4. Set $u_{k+1} = u_k + \alpha_k d_k$;
    5. Set $k = k+1$;
    6. End while
```

Convergence of the steepest descent scheme is established with the largest steplength $\alpha_k \in \{\alpha^\ell, \ell = 0,1,\ldots\}$, $\alpha \in (0,1)$, that satisfies the Armijo condition of sufficient decrease of $\hat{J}$'s value given by

$$\hat{J}(u_k + \alpha_k d_k) \leq \hat{J}(u_k) + \delta\,\alpha_k\,(\nabla\hat{J}(u_k), d_k)_U \tag{4.4}$$

together with the Wolfe condition

$$(\nabla\hat{J}(u_k + \alpha_k d_k), d_k)_U > \sigma\,(\nabla\hat{J}(u_k), d_k)_U, \tag{4.5}$$

where $0 < \delta < \sigma < 1/2$; see [265]. The last condition means that the graph of $\hat{J}$ should not increase too fast beyond the minimum. Notice that we use the inner product of the U space.

NCG schemes represent extensions of linear conjugate gradient methods to non-quadratic problems; see, e.g., [319, 145]. In the common variants, the basic idea is to avoid matrix operations and express the search directions recursively as

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \tag{4.6}$$

where $g_k = \nabla \hat{J}(u_k)$, $k = 0,1,2,\ldots$, with $d_0 = -g_0$. The iterates for a minimum point are given by

$$u_{k+1} = u_k + \alpha_k\, d_k, \tag{4.7}$$

where $\alpha_k > 0$ is a steplength. The parameter β_k is chosen so that (4.6)–(4.7) reduces to the linear CG scheme if $\hat{J}$ is a strictly convex quadratic function and α_k is the exact one-dimensional minimizer of $\hat{J}$ along d_k. In this case the NCG scheme terminates in at most n steps in exact arithmetic. This case provides a lower bound to the computational complexity of NCG schemes.

There are many different formulas for β_k which result in different performances depending on the (nonlinear) problem. Well-known formulas are the following:

Fletcher–Reeves, $\beta^{FR} = \frac{(g_{k+1}, g_{k+1})_U}{(g_k, g_k)_U}$;

Polak–Ribière, $\beta^{PR} = \frac{(g_{k+1}, g_{k+1}-g_k)_U}{(g_k, g_k)_U}$;

Hestenes–Stiefel, $\beta^{HS} = \frac{(g_{k+1}, g_{k+1}-g_k)_U}{(d_k, g_{k+1}-g_k)_U}$.

A more recent formulation is due to Dai and Yuan [107] as follows

$$\beta_k^{DY} = \frac{(g_{k+1}, g_{k+1})_U}{(d_k, y_k)_U},$$

where $y_k = g_{k+1} - g_k$, and another one is due to Hager and Zhang [181] based on the formula

$$\beta_k^{HZ} = \frac{(\sigma_k, g_{k+1})_U}{(d_k, y_k)_U}, \qquad \sigma_k = y_k - 2 d_k \frac{(y_k, y_k)_U}{(y_k, d_k)_U}. \tag{4.8}$$

Results in [63, 354] suggest that the last two formulations are advantageous in PDE optimization. We note the definition of β_k based on U-space inner product. For optimization in complex Hilbert spaces, the inner products above are replaced with $\Re e(\cdot,\cdot)_U$; see [63].

The NCG scheme is implemented as follows.

ALGORITHM 4.3. NCG scheme.

```
• Input: initial approx. $u_0$, $d_0 = -\nabla \hat{J}(u_0)$, index $k = 0$, maximum $k_{max}$, tolerance $tol$.
    1. While ($k < k_{max}$ && $\|g_k\|_U > tol$ ) do
    2. Evaluate steplength $\alpha_k > 0$ along $d_k$ satisfying (4.13)–(4.14);
    3. Set $u_{k+1} = u_k + \alpha_k\, d_k$;
    4. Compute $g_{k+1} = \nabla \hat{J}(u_{k+1})$;
    5. Compute $\beta_k$ by, e.g., (4.8);
    6. Let $d_{k+1} = -g_{k+1} + \beta_k\, d_k$;
    7. Set $k = k + 1$;
    8. End while
```

4.2.2 Quasi-Newton Methods

Quasi-Newton methods can be seen as extensions of the conjugate gradient method, in which additional storage is used to accelerate convergence. In these methods, approximations of the Hessian matrix are constructed using low-rank updates based on gradient evaluations.

The BFGS method is a quasi-Newton method which makes successive rank-two updates to a matrix B such that it serves as an approximation to the true Hessian. Typically, the BFGS scheme exhibits convergence rates superior to those of NCG schemes at the expense of additional computational effort.

Denote with B_k the kth BFGS approximation to the Hessian. Then the BFGS search direction at the kth step is given by $p_k = -B_k^{-1} g_k$. Further, denote the difference between two successive updates of u as $s_k = \alpha_k p_k$, where α_k is the steplength. The matrix B can be formed explicitly via the well-known recurrence formula

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)^\top}{s_k^\top B_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}. \tag{4.9}$$

To compute the search direction, it is necessary to invert the matrix B. We denote its inverse as $H = B^{-1}$. Using the Sherman–Morrison–Woodbury formula, we can also establish a recurrence for H

$$H_{k+1} = H_k + \frac{s_k^\top y_k + y_k^\top H_k y_k}{(s_k^\top y_k)^2}(s_k s_k^\top) - \frac{H_k y_k s_k^\top + s_k y_k^\top H_k}{s_k^\top y_k}. \tag{4.10}$$

In the case where the control u and the gradient of the objective function $\nabla \hat{J}$ are elements in a function space, it is not immediately obvious how to directly use this formula since it requires forming outer products. Moreover, to compute the search direction, we only need the action of H on a vector g and it is not necessary to construct any matrix. These facts are discussed in [354], where a matrix-free BFGS is formulated; see also [248, 264, 252].

Suppose U is either $L^2(0,T;\mathbb{R})$ or $H^1(0,T;\mathbb{R})$ and that x, $y \in U$. Then we can denote the function space analogue of the outer product as a dyadic operator $x \otimes y : U \to U$. The action of this operator on a third element $z \in U$ can be expressed in terms of the inner product $(x \otimes y) z = (y,z)_U \, x$.

Now, we illustrate the computational steps to compute the BFGS solution. We start with the initial approximation u_0 and correspondingly determine $g_0 = \nabla \hat{J}(u_0)$. We set $p_0 = -g_0$ and minimize along p_0 with line search with steplength α_0. We obtain u_1 and g_1, and therefore we can compute $y_0 = g_1 - g_0$ and $s_0 = \alpha_0 \, p_0$. The first step requires the initialization $H_0 = I$, and we have $z_0 = H_0 \, y_0$ and $p_1 = -H_0 \, g_1$. Correspondingly, minimizing along p_1 with line search with steplength α_1, we obtain $y_1 = g_2 - g_1$ and $s_1 = \alpha_1 \, p_1$.

From the recursion relation for H, we obtain the following summation formula for the search direction for $k > 0$. The term $z_k = H_k \, y_k$ is computed as follows

$$z_k = H_0 \, y_k + \sum_{j=1}^{k-1} \left[c_j \, (s_j, y_k)_U \, r_j - (z_j, y_k)_U \, s_j \right], \tag{4.11}$$

where $c_j = (s_j, y_j)_U^{-1}$, $d_j = 1 + c_j\,(y_j, z_j)_U$, and $r_j = d_j\,s_j - z_j$. The BFGS search directions are computed as follows

$$p_{k+1} = -H_0\,g_{k+1} - \sum_{j=1}^{k} \left[c_j\,(s_j, g_{k+1})_U\,r_j - (z_j, g_{k+1})_U\,s_j \right]. \tag{4.12}$$

In the numerical implementation, these functions are approximated on a grid and the L^2- and H^1-inner products are in either case approximated by a vector inner product with a weighting matrix. This can be written as $(u,v)_{L^2} \approx u^\top M v$ and $(u,v)_{H^1} \approx u^\top K v$, where M is the mass matrix and K is the stiffness matrix.

Notice that with both the NCG and BFGS schemes, the new control u_{k+1} is composed of a linear combination of the original control and the gradients at every step. Further, in the BFGS approach the current approximation to the inverse of the Hessian is stored, whereas in the matrix-free BFGS method the vectors $\{s_j, y_j, z_j\}$ are stored. These are the s_j vectors which are the search steps themselves, the y_j which are the differences between successive gradients, and the z_j vectors which are elements in the space spanned by $\{s_0, \ldots, s_{j-1}\}$. As a counterpart, the matrix-free BFGS formula requires progressively more computation for each optimization step, so it is important that the improved convergence properties at least compensate for the increased computational effort. The matrix-free BFGS algorithm is given below.

ALGORITHM 4.4. BFGS scheme.

```
• Input: choose $H_0 = I$, initial approx. $u_0$, $g_0 = \nabla \hat{J}(u_0)$, $p_0 = -g_0$, index $k = 0$,
  maximum $k_{max}$, tolerance $tol$.

• Compute $u_1 = u_0 + \alpha_0\,p_0$ with $\alpha_0$ satisfying (4.13)–(4.14); compute $g_1 = \nabla \hat{J}(u_1)$,
  $y_0 = g_1 - g_0$, $s_0 = \alpha_0\,p_0$, and $p_1 = -H_0\,g_1$;

• Set $k = 1$;

    1. While ( $k < k_{max}$ && $\|g_{k-1}\|_U > tol$ ) do
    2. Compute $u_{k+1} = u_k + \alpha_k\,p_k$ with $\alpha_k$ satisfying (4.13)–(4.14);
    3. Compute $g_{k+1} = \nabla \hat{J}(u_{k+1})$, $y_k = g_{k+1} - g_k$, $s_k = \alpha_k\,p_k$;
    4. Compute $z_k$ with (4.11);
    5. Compute and save $c_k = (s_k, y_k)_U^{-1}$, $d_k = 1 + c_k\,(y_k, z_k)_U$, and $r_k = d_k\,s_k - z_k$;
    6. Compute new search direction $p_{k+1}$ with (4.12).
    7. Set $k = k + 1$;
    8. End while
```

Note that the typically superlinear convergence properties of quasi-Newton methods are in Hilbert spaces obtained only if the initial Hessian approximation is chosen as a compact perturbation of the Hessian at the optimal solution (cf. [228, 297, 153]).

4.2.3 Krylov–Newton Methods

Newton methods aim at solving the necessary first-order optimality condition that the gradient $\nabla \hat{J}(u)$ vanish at a local minimum. The Newton method consists of the following iterative procedure

$$\nabla^2 \hat{J}(u_k)\,\delta u = -\nabla \hat{J}(u_k),$$
$$u_{k+1} = u_k + \delta u,$$

where $\nabla^2 \hat{J}(u_k)$ denotes the reduced Hessian. Because of the typical large size of PDE optimization problems, the Newton equation is solved iteratively using a Krylov method. This solution process involves solving PDE problems that can be done reasonably only to a given tolerance. This means that Newton methods applied to PDE optimization problems are always inexact [111]. Moreover, due to discretization and other approximation introduced in the computation of the Hessian and of the gradient, we should refer to inexact perturbed Newton methods. See [102] for a related discussion and the investigation of the natural connection between quasi-Newton, inexact, and inexact perturbed Newton methods.

In the following, we discuss the Krylov–Newton scheme that is a particular instance of the inexact Newton method. In order to anticipate the possible lack of positive definiteness and still exploit the symmetry of the Hessian, we use the Krylov-type symmetric LQ (SYMMLQ) method [130]. In our experience, this method robustly computes search directions in less time than other Krylov methods, such as GMRES or BiCG, if no preconditioner or a symmetric preconditioner is used. If the Hessian has negative eigenvalues, the SYMMLQ scheme may compute an ascent direction. Whether the direction δu is an ascent or descent can be determined from the sign of its projection onto the gradient. In the cases where δu is an ascent direction, we use $-\delta u$ as a descent direction.

Notice that we refer to δu as the descent direction and not as increment of the optimization variable. In fact, the objective may be nonconvex and the constraint nonlinear, and therefore the approximate solution of the Newton equation cannot guarantee a correct update. Robustness of the Newton approach is obtained using a globalization strategy with a robust linesearch procedure.

In this section, we illustrate a globalized Krylov–Newton scheme; see [355, 356] for theoretical and implementation details. The workflow of the optimization scheme is given by Algorithm 4.5, which requires the computation of the Newton descent direction given by Algorithm 4.7 and a linesearch scheme. A MATLAB implementation of this solution procedure for solving optimal quantum control problems is given in [355].

ALGORITHM 4.5. Optimization scheme.

```
• Input: initial approx. $u_0$, index $k = 0$, maximum $k_{max}$, tolerance $tol$.
    1. While ( $k < k_{max}$ && $\|\nabla \hat{J}(u_k)\|_U > tol$ ) do
    2. Compute search direction $\delta u$ with Algorithm 4.7;
    3. Compute $\alpha$ with a linesearch algorithm;
    4. Set $u_{k+1} = u_k + \alpha\,\delta u$;
    5. Set $k = k + 1$;
    6. End while
```

ALGORITHM 4.6. Apply the reduced Hessian to δu.

```
• At $(y, u, p)$ and $\delta u$ given
    1. Solve $c_y\,\delta y = -c_u\,\delta u$;
    2. Solve $c_y^*\,\delta p = -[L_{yy}\,\delta y + L_{yu}\,\delta u]$;
    3. Assemble $\nabla^2 \hat{J}(u)\,\delta u := L_{uu}\,\delta u + c_u^*\,\delta p + L_{uy}\,\delta y$;
    4. End
```

ALGORITHM 4.7. Solve the Hessian problem.

```
• Given $u$ and an initial guess to $\delta u$, e.g., $\delta u = -L_{uu}^{-1}\nabla\hat{J}(u)$
    1. Iteratively solve $\nabla^2\hat{J}(u)\,\delta u = -\nabla\hat{J}(u)$ with SYMMLQ (Use Algorithm 4.6 to
       apply the Hessian);
    2. If ascent direction $(\delta u, \nabla\hat{J}(u))_U > 0$ set $\delta u := -\delta u$;
    3. End
```

With a Newton method, it is recommended to begin the line search with $\alpha = 1$; see, e.g., [265]. This is a reasonable choice when the functional is locally quadratic and knowing that the Newton update is automatically scaled in the functional space where the Hessian and the gradient are defined. However, in nonconvex optimization problems the desired steplength can be orders of magnitude smaller. Therefore, it is advantageous to have an upper bound α_{max} on the maximum feasible steplength. See [356] for details on this issue and the use of α_{max} to design a robust bisection linesearch scheme. The usage of left-sided block-preconditioners for GMRES in constrained problems is illustrated in Section 4.4.

4.2.4 Cascadic Black-Box Schemes

The cascadic approach results from combining nested iteration techniques with single-grid solvers. To illustrate this approach, consider a hierarchy of grids with index $lev = lev_0, \ldots, lev_f$. The idea is to start from a coarse grid with index lev_0, where the problem is small-sized and still well represented. On this grid, the PDE optimization problem can be solved by an iterative black-box scheme with a small computational effort. Let us denote with x_{lev_0} the solution obtained by this process with initialization given by $x^*_{lev_0}$. The step that follows is to interpolate the solution x_{lev_0} to the next finer grid, using an interpolation operator I_{lev}^{lev+1}. Therefore we obtain an initialization of the black-box iterative process on the finer grid that is given by

$$x^*_{lev+1} = I_{lev}^{lev+1}\, x_{lev},$$

where $lev = lev_0$. With this initialization and using the black-box scheme we obtain the solution x_{lev+1}. This process is repeated until the finest grid is reached and the desired solution is obtained. This method is summarized in Algorithm 4.8, where we denote with $x_{lev} = BB_{lev}(x^*_{lev})$ the result of the black-box iteration, with x^*_{lev} as initialization.

ALGORITHM 4.8. Cascadic method.

```
• Given $lev = lev_0$ and coarse initialization $x^*_{lev_0}$.
    1. Compute $x_{lev} = BB_{lev}(x^*_{lev})$;
    2. If $lev = lev_f$ then stop;
    3. Else if $lev < lev_f$ then interpolate $x^*_{lev+1} = I^{lev+1}_{lev} x_{lev}$;
    4. Set $lev = lev + 1$, goto 1.;
    5. End
```

The original motivation for using the cascadic approach comes from computational experience and the results given in [43, 318], where a cascadic conjugate gradient method is discussed and optimal computational complexity for elliptic problems is proved. See, e.g., [63, 354] for successful implementation of the cascadic scheme with NCG and BFGS schemes.

Notice that we have discussed black-box methods for PDE optimization problems without constraints on the optimization variable. In the case where the admissible set $U_{ad} \subset U$ is a closed convex subset of the optimization space, a projection operator $\mathcal{P}_{U_{ad}}(u)$ on U_{ad} of the updates of u should be applied. In particular, an update step becomes

$$u_{k+1} = \mathcal{P}_{U_{ad}}(u_k + \alpha_k d_k),$$

where d_k is the kth optimization direction and α_k is obtained with a linesearch procedure which includes the projection. Specifically, the Armijo condition of sufficient decrease of $\hat{J}$'s value is given by

$$\hat{J}(\mathcal{P}_{U_{ad}}(u_k + \alpha_k d_k)) \leq \hat{J}(u_k) + \delta\, \alpha_k\, (\nabla \hat{J}(u_k), \tilde{d}_k)_U \tag{4.13}$$

and the Wolfe condition becomes

$$(\nabla \hat{J}(\mathcal{P}_{U_{ad}}(u_k + \alpha_k d_k)), d_k)_U > \sigma\, (\nabla \hat{J}(u_k), \tilde{d}_k)_U, \tag{4.14}$$

where $0 < \delta < \sigma < 1/2$, and $\tilde{d}_k$ is as follows

$$\tilde{d}_k = \begin{cases} d_k, & u_k \in int(U_{ad}), \\ 0, & u_k \in \partial U_{ad} \text{ and } d_k \text{ points outwards } U_{ad}. \end{cases}$$

4.3 Semismooth Newton Methods

In this section, we briefly discuss an extension of the Newton method to the case of PDE optimization problems where additional constraints to the optimization variables result in variational inequalities in the optimality system. In particular, we consider constraints such that the KKT operator equation is Lipschitz continuous but not C^1 regular. For a detailed discussion on this topic and for references see [207].

To illustrate the case of Lipschitz continuous but not C^1 regular function, we consider the following complementarity problem

$$g(u) \leq 0, \qquad u \leq \overline{u}, \qquad g(u)^\top (u - \overline{u}) = 0, \tag{4.15}$$

where $g : \mathbb{R}^n \to \mathbb{R}^n$ and $\overline{u} \in \mathbb{R}^n$ and the inequalities must be interpreted componentwise. The solution to this problem can be formulated as the solution to the following equation

$$F(u) := g(u) + \max(0, -g(u) + u - \overline{u}) = \max(g(u), u - \overline{u}) = 0.$$

If g is locally Lipschitz and C^1 regular, it results that F is a locally Lipschitz continuous function, but it is not C^1. A function is called locally Lipschitz continuous if it is Lipschitz continuous on every bounded subset of its domain.

Next, consider the following elliptic distributed optimal control problem

$$\left\{\begin{array}{rcll} \min_{u\in U_{ad}} J(y,u) & := & \frac{1}{2}\|y-z\|^2_{L^2(\Omega)} + \frac{\nu}{2}\|u\|^2_{L^2(\Omega)}, & \\ -\Delta y & = & u+g & \text{in } \Omega, \\ y & = & 0 & \text{on } \partial\Omega, \end{array}\right. \tag{4.16}$$

where we require that the set of admissible controls be the closed convex subset of $L^2(\Omega)$ given by

$$U_{ad} = \{u \in L^2(\Omega) \,|\, \underline{u}(\mathbf{x}) \le u(\mathbf{x}) \le \overline{u}(\mathbf{x}) \text{ a.e. in } \Omega\}, \tag{4.17}$$

where $\underline{u}$ and $\overline{u}$ are elements of $L^\infty(\Omega)$.

We have that the solution is characterized by the following optimality system

$$\begin{array}{rll} -\Delta y = & u+g & \text{in } \Omega, \\ y = & 0 & \text{on } \partial\Omega, \\ -\Delta p = & -(y-z) & \text{in } \Omega, \\ p = & 0 & \text{on } \partial\Omega, \\ (\nu u - p, v-u) \ge & 0 & \forall v \in U_{ad}. \end{array} \tag{4.18}$$

We see that in the presence of control constraints $\underline{u} \le u \le \overline{u}$, the optimality condition $\nabla \hat{J}(u) = 0$ is replaced by the variational inequality $(\nabla \hat{J}(u), v-u) \ge 0$ for all admissible $v \in U_{ad}$. This condition can be reformulated introducing Lagrange multipliers as follows

$$\nabla \hat{J}(u) + \lambda_H - \lambda_L = 0, \tag{4.19}$$

$$\lambda_L \ge 0, \qquad \underline{u} - u \le 0, \qquad (\lambda_L, \underline{u} - u) = 0, \tag{4.20}$$

$$\lambda_H \ge 0, \qquad u - \overline{u} \le 0, \qquad (\lambda_H, u - \overline{u}) = 0. \tag{4.21}$$

Also in this case, we can reformulate (4.19) and the pointwise (a.e. in Ω) complementarity problems (4.20) and (4.21) using max and min functions as follows

$$\nabla \hat{J}(u) + \lambda = 0, \tag{4.22}$$

$$\max\{0, \lambda + c\,(u - \overline{u})\} + \min\{0, \lambda + c\,(u - \underline{u})\} - \lambda = 0, \tag{4.23}$$

where $\lambda = \lambda_H - \lambda_L$ and any $c > 0$.

Notice that the operator equation (4.23) is Lipschitz continuous but not differentiable in the classical sense. However, in the case $c = \nu$ it enjoys a property called Newton differentiability that generalizes the concept of differentiability to a.e. differentiable Lipschitz continuous functions; see, e.g., [207].

To illustrate this concept in the finite-dimensional case, consider a locally Lipschitz continuous function $F : \mathbb{R}^n \to \mathbb{R}^n$ and let D_F denote the set of points at which F is differentiable. For $u \in \mathbb{R}^n$ we define $\partial_B F(x)$ as

$$\partial_B F(u) = \left\{ d : \lim_{u_i \to u,\, u_i \in D_F} \nabla F(u_i) \right\}.$$

Here, B stands for Bouligand differentiable. We can state that a locally Lipschitz function F is B-differentiable at u if and only if it is directionally differentiable at this point. Further, we denote with $\partial F(u)$ the generalized derivative at u defined as the convex hull $\partial F(u) = co\, \partial_B F(u)$; see [104]. Then the directional derivative $F'(u;\delta u)$ is Lipschitz continuous, and there exists a $\tilde{V}(u) \in \partial F(u)$ such that

$$F'(u;\delta u) = \tilde{V}(u)\,\delta u.$$

In this case, F is said to be semismooth at u. Furthermore, for every $V(u) \in \partial F(u)$, we have

$$V(u)\,\delta u - F'(u;\delta u) = o(|\delta u|) \qquad \text{as } \delta u \to 0.$$

Now, we can discuss a generalized Newton iteration for solving $F(u) = 0$. We have

$$u_{k+1} = u_k - V_k^{-1} F(u_k), \qquad V_k = V(u_k) \in \partial_B F(u_k). \tag{4.24}$$

For example, consider the complementarity problem discussed at the beginning of this section. We have $F(u) = g(u) + \max(0, -g(u) + u - \overline{u})$, and the corresponding semismooth derivative results as follows

$$V(u) = \begin{cases} 1 & \text{if} \quad -g(u) + u - \overline{u} > 0, \\ g'(u) & \text{if} \quad -g(u) + u - \overline{u} \le 0, \end{cases} \tag{4.25}$$

and this definition must be interpreted componentwise.

We remark that the result (4.25) is representative of an important fact. We have that a semismooth Newton iteration can be equivalently formulated as a primal-dual active set method [191]. In fact, consider the semismooth Newton iteration (4.24) and define the following active and inactive sets (componentwise)

$$\mathcal{A} = \{i : -g(u)(i) + u(i) - \overline{u}(i) > 0\} \qquad \text{and} \qquad \mathcal{I} = \{i : -g(u)(i) + u(i) - \overline{u}(i) \le 0\},$$

where $i = 1, \ldots, n$. On these sets, we obtain the δu increment as the solution to the following

$$\delta u + u - \overline{u} = 0 \text{ on } \mathcal{A} \qquad \text{and} \qquad g'(u)\,\delta u + g(u) = 0 \text{ on } \mathcal{I}.$$

We see that the u update obtained with these increments corresponds to the semismooth Newton update (4.24)–(4.25).

Next, we discuss the Newton method with a primal-dual active set strategy. For this purpose, we rewrite the complementarity problem (4.15), in terms of primal and dual variables, as $P(u,\lambda) = 0$, where

$$P(u,\lambda) = \begin{pmatrix} g(u) + \lambda \\ \max\{0, \lambda + c\,(u - \overline{u})\} - \lambda \end{pmatrix}, \tag{4.26}$$

where $c > 0$.

One step of the primal-dual active set strategy applied to this problem proceeds as follows

$$\begin{aligned} &\text{Solve} \quad g(u_{k+1}) + \lambda_{k+1} = 0 \qquad \text{with} \\ &\quad u_{k+1} = \overline{u} \qquad \text{in } \mathcal{A} = \{i : \lambda_k(i) + c\,(u_k(i) - \overline{u}(i)) > 0\}, \\ &\quad \lambda_{k+1} = 0 \qquad \text{in } \mathcal{I} = \{i : \lambda_k(i) + c\,(u_k(i) - \overline{u}(i)) \le 0\}. \end{aligned}$$

On the other hand, a semismooth Newton step on the same problem is given by

$$\begin{array}{lll} \text{Solve} & g'(u_k)(u_{k+1}-u_k)+g(u_k)+\lambda_{k+1}=0 & \text{with} \\ & u_{k+1}=\overline{u} \quad \text{in } \mathcal{A}=\{i:\lambda_k(i)+c\,(u_k(i)-\overline{u}(i))>0\}, & \\ & \lambda_{k+1}=0 \quad \text{in } \mathcal{I}=\{i:\lambda_k(i)+c\,(u_k(i)-\overline{u}(i))\le 0\}. & \end{array}$$

Now, we focus on (4.22)–(4.23) representing the elliptic distributed optimal control problem with bilateral control constraints. The two active sets and one inactive set at the kth Newton iteration are given by

$$\mathcal{A}_k^L=\{x\in\Omega:\lambda_k(x)+c\,(u_k(x)-\underline{u}(x))<0\}$$

and

$$\mathcal{A}_k^H=\{x\in\Omega:\lambda_k(x)+c\,(u_k(x)-\overline{u}(x))>0\},$$

and $\mathcal{A}_k=\mathcal{A}_k^L\cup\mathcal{A}_k^H$, whereas the inactive set $\mathcal{I}_k=\Omega\setminus\mathcal{A}_k$. With this setting, a semismooth Newton step is formulated as follows

$$\left[\begin{array}{cc} \nabla^2\hat{J}(u_k) & \chi_{\mathcal{A}_k} \\ \chi_{\mathcal{A}_k} & 0 \end{array}\right]\left(\begin{array}{c} \delta u \\ \delta\lambda_{\mathcal{A}_k} \end{array}\right)=-\left(\begin{array}{c} \nabla\hat{J}(u_k)+\lambda_k \\ \chi_{\mathcal{A}_k^L}(u_k-\underline{u})+\chi_{\mathcal{A}_k^H}(u_k-\overline{u}) \end{array}\right), \tag{4.27}$$

where $\chi_{\mathcal{U}}$ denotes the characteristic function of a set $\mathcal{U}$, and $\delta\lambda_{\mathcal{A}_k}$ denotes the restriction of $\delta\lambda$ to the active set $\mathcal{A}_k$.

We summarize the semismooth Newton iteration in the following algorithm.

ALGORITHM 4.9. Newton method with primal-dual active set strategy.

```
• Input: initial approx. $u_0$ and $\lambda_0$, index $k=0$, maximum $k_{max}$.
    1. While ( $k<k_{max}$ && $\mathcal{A}_k\neq\mathcal{A}_{k+1}$ ) do
    2. Evaluate $\nabla\hat{J}(u_k)$;
    3. Determine the active/inactive sets, $\mathcal{A}_k^L$, $\mathcal{A}_k^H$, $\mathcal{A}_k$, and $\mathcal{I}_k$;
    4. Assemble the right-hand side of the Newton equation;
    5. Solve the Newton equation (4.27) iteratively;
    6. Set $u_{k+1}=u_k+\delta u$, $\lambda_{k+1}=u_k+\delta u$ on $\mathcal{A}_k$, and $\lambda_{k+1}=0$ on $\mathcal{I}_k$ (globalization
       may be required);
    7. Set $k=k+1$;
    8. End while
```

Next, we reformulate the optimality system (4.18) in view of the preceding discussion and formulate a primal-dual iteration on this system which represents a realization of the Newton iteration described above. We have

$$\begin{array}{rcl} -\Delta y &=& u+g, \\ y &=& 0, \\ -\Delta p &=& -(y-z), \\ p &=& 0, \\ \nu u-p+\lambda &=& 0, \\ \max\{0,\lambda+c\,(u-\overline{u})\}+\min\{0,\lambda+c\,(u-\underline{u})\}-\lambda &=& 0. \end{array}$$

ALGORITHM 4.10. Primal-dual active set strategy for bilateral control constraints.

```
• Input: initial approx. $u_0$ and $\lambda_0$, index $k = 0$, maximum $k_{max}$, $c > 0$.
    1. While ( $k < k_{max}$ && $\mathcal{A}_k \neq \mathcal{A}_{k+1}$ ) do
    2. Determine the active/inactive sets
           $\mathcal{A}_k^L = \{x \in \Omega : \lambda_k(x) + c\,(u_k(x) - \underline{u}(x)) < 0\}$
       and
           $\mathcal{A}_k^H = \{x \in \Omega : \lambda_k(x) + c\,(u_k(x) - \overline{u}(x)) > 0\}$,
       and $\mathcal{A}_k = \mathcal{A}_k^L \cup \mathcal{A}_k^H$, $\mathcal{I}_k = \Omega \setminus \mathcal{A}_k$.
    3. Solve the following system
           $-\Delta y_k = f_k + g$,
           $y_k = 0$,
           $-\Delta p_k = -(y_k - z)$,
           $p_k = 0$,
       where
           $f_k(x) = \begin{cases} \underline{u}(x) & \text{if } x \in \mathcal{A}_k^L, \\ p_k(x)/\nu & \text{if } x \in \mathcal{I}_k, \\ \overline{u}(x) & \text{if } x \in \mathcal{A}_k^H. \end{cases}$
    4. Set $\lambda_{k+1} = p_k - \nu\, u_k$;
    5. Set $k = k + 1$;
    6. End while
```

For further discussion on semismooth Newton schemes and applications, we refer the reader to [207]. A case of polygonal constraints is discussed in [226]. Convex control constraints are discussed in [357]. A case of a problem with control constraints with L^1 cost solved with a semismooth Newton scheme is discussed in [323].

4.4 Preconditioning

A generic problem class in PDE constrained optimization is defined by linear-quadratic problems. Either they arise as such or as a subproblem within SQP methods. In any case, the solution of this type of problem is important, and if it is not performed by multigrid methods, most often Krylov subspace methods are employed, like GMRES, or, using the specific symmetric problem structure, SYMMLQ and MINRES. Because of the spread-out eigenvalue structure of quadratic programming (QP) problems in PDE constrained optimization, we expect rather slow convergence of those Krylov subspace methods. Therefore, preconditioning techniques have to be employed in order to speed up convergence. The general guideline for preconditioning is to try to squeeze the eigenvalues of the preconditioned systems into a small number of tight clusters, whereas the action of the preconditioner itself should be as cheap as possible. In particular, we focus on the following generic quadratic optimization problem with linear equality constraints

$$\begin{aligned}&\min_{y,u} \tfrac{1}{2}\left(y^\top H_{yy}\, y + y^\top H_{yu}\, u + u^\top H_{uy}\, y + u^\top H_{uu}\, u\right) + f_y^\top x + f_u^\top u,\\ &C_y\, y + C_u\, u + c = 0,\end{aligned} \tag{QP}$$

where $y \in \mathbb{R}^{n_y}$, $u \in \mathbb{R}^{n_u}$ are the variable vectors of the optimization problem, the vectors $c, f_y \in \mathbb{R}^{n_y}$, $f_u \in \mathbb{R}^{n_u}$, and the matrices arising are of consistent dimensions:

$$\begin{aligned}&H_{yy} \in \mathbb{R}^{n_y \times n_y}, \quad H_{uu} \in \mathbb{R}^{n_u \times n_u}, \quad H_{yu} = H_{uy}^\top \in \mathbb{R}^{n_y \times n_u},\\ &C_y \in \mathbb{R}^{n_y \times n_y}, \quad C_u \in \mathbb{R}^{n_y \times n_u}.\end{aligned}$$

We assume as an important structural property that C_y is nonsingular and utilize a nullspace basis Z such that $[C_y\ C_u]Z = 0$:

$$Z = \begin{bmatrix} -C_y^{-1} C_u \\ I \end{bmatrix}.$$

Considering the Hessian of (QP), we have

$$H := \begin{bmatrix} H_{yy} & H_{yu} \\ H_{uy} & H_{uu} \end{bmatrix},$$

and we assume that the respective reduced Hessian

$$S = Z^\top H Z = H_{uu} - H_{uy} C_y^{-1} C_u - C_u^\top C_y^{-\top} H_{yu} + C_u^\top C_u^{-\top} H_{yy} C_y^{-1} C_u \tag{4.28}$$

is positive definite, which guarantees the unique solvability of (QP). These linear-quadratic problems typically arise as subproblems within large-scale model-based optimization tasks, where the constraint consists of a discretized PDE. In the following, we focus on the linear-quadratic problem itself.

Problem (QP) is known to satisfy a KKT system of the form

$$\begin{bmatrix} H_{yy} & H_{yu} & C_x^\top \\ H_{uy} & H_{uu} & C_p^\top \\ C_y & C_u & 0 \end{bmatrix} \begin{pmatrix} y \\ u \\ p \end{pmatrix} = - \begin{pmatrix} f_y \\ f_u \\ c \end{pmatrix}, \tag{4.29}$$

and we should note that the reduced Hessian S can be interpreted as the Schur complement of the KKT matrix in (4.29) with respect to the variables (y, p). The block structure in the system matrix has motivated research in block-structured preconditioners for Krylov subspace methods. In [26] three different preconditioners for the solution of (QP) with MINRES and SYMMLQ [271] are analyzed and tested. In [37] several preconditioners based on reduced space techniques are investigated and applied to accelerate GMRES [295] iterations for the linear system. In [162] again a block-structured preconditioner is used to accelerate QMR [130] for model-based output least-squares problems. The article [114] again studies approximate block factorizations in the context of Krylov iterations.

Equation (4.29) is a saddle-point problem with specific features. The only natural assumptions for this system are the assumption that C_y is invertible and that the reduced Hessian in (4.28) is positive definite. These assumptions result from the fact that (4.29) is derived from a PDE-constrained optimization problem. Often, (4.29) is compared with variational optimization problems, like the Stokes problem. The key feature of variational optimizations problem is, however, the positivity of the $\begin{bmatrix} H_{yy} & H_{yu} \\ H_{uy} & H_{uu} \end{bmatrix}$-block. This is essential also for preconditioning methods. Although the typical test problem also possesses this property, we will not discuss any further methods which rely on the positivity of this block, since there are many meaningful PDE-constrained optimization problems with rank deficient $\begin{bmatrix} H_{yy} & H_{yu} \\ H_{uy} & H_{uu} \end{bmatrix}$-block. In [285] so-called optimal preconditioners are presented which rely on multigrid iterations for subblocks and also assume the positivity of the Hessian block.

Readers interested in an overview of various methods for preconditioning of optimization problems are referred to [32, 64]. In this section, we focus on preconditioning methods, which are considered as highly successful by many authors [37, 26, 211, 122] for the particular problem class under investigation.

If one aims at preserving the symmetry structure in (4.29), which we denote with $Kz = r$, one is searching for inexpensively invertible linear transformations P such that the system matrix in

$$P^{-1} K P^{-\top} \tilde{z} = P^{-1} r \qquad (\tilde{z} = P^{\top} z) \tag{4.30}$$

is still symmetric and possesses clustered eigenvalues. Thus, SYMMLQ or MINRES can be applied to the preconditioned equation (4.30). Obviously, symmetric preconditioners have to involve roots of arising operators and therefore mostly rely on positive definiteness of the Hessian part. Thus, typically they are not focused on the most reliable part of the KKT system, which is the PDE system matrix, which has to be invertible and of which in most cases good approximations are known. Also, in [188] a promising indefinite preconditioner for a projected preconditioned conjugate gradient method is introduced which is still independent of the PDE system operator.

Those preconditioners, which take into account available knowledge of the PDE system operator, are of the form of a left-sided preconditioner

$$P_L^{-1} K z = P_L^{-1} r. \tag{4.31}$$

The resulting preconditioned system is no longer symmetric. Therefore, general Krylov subspace methods like typically GMRES are applied. Most authors (see, e.g., [37, 26, 211, 122]) agree on the following form of this left-sided preconditioner which can be derived from reduced SQP methods or nullspace factorizations

$$P_L = \begin{bmatrix} 0 & 0 & A^{\top} \\ 0 & B & C_p^{\top} \\ A & C_p & 0 \end{bmatrix}, \tag{4.32}$$

where A is an approximation to the forward system matrix and B is an approximation to the reduced Hessian. In [26, 211], it is observed that the choice $A = C_y$ and $B = S$ results in a nilpotent iteration matrix $I - P_L^{-1} K$ of nilpotency degree 3, and thus a Krylov subspace method converges within three steps in this case. The results in [211, 122] indicate that the matrix B should be an approximation to the approximate reduced Hessian S_A,

$$S_A = H_{uu} - H_{uy} A^{-1} C_{uu} - C_{uu}^{\top} A^{-\top} H_{yu} + C_{uu}^{\top} A^{-\top} H_{yy} A^{-1} C_{uu}, \tag{4.33}$$

rather than to the exact reduced Hessian (4.28). This should be kept in mind if the reduced Hessian approximation is generated by quasi-Newton-like update strategies. On the other hand, analytic information can often be obtained for the exact reduced Hessian. And this information can be profitably used as an approximation to S_A.

Next, we illustrate the application of the preconditioning technique discussed above to the basic elliptic control problem. Consider

$$\min \frac{1}{2}\int_\Omega (y(\xi)-\bar{y}(\xi))^2 d\xi + \frac{\nu}{2}\int_\Omega u(\xi)^2 d\xi$$
$$-\triangle y(\xi) = u(\xi) \quad \forall \xi \in \Omega,$$
$$y(\xi) = 0 \quad \forall \xi \in \partial\Omega.$$

The variables y and u are functions defined on the domain Ω, and $\triangle$ denotes the Laplacian operator. The aim of the problem is to track the given target function $\bar{y}$. We choose $\Omega = [0,1]$. Then, this model problem simplifies to

$$\min \frac{1}{2}\int_0^1 (y(\xi)-\bar{y}(\xi))^2 d\xi + \frac{\nu}{2}\int_0^1 u(\xi)^2 d\xi$$
$$-y''(\xi) = u(\xi) \quad \forall \xi \in [0,1],$$
$$y(0)=0,\, y(1)=0.$$

The function $\bar{y}$ is chosen as (cf. Figure 4.1)

$$\bar{y}(\xi) = \begin{cases} 0.8-\xi, & 0 \le \xi \le 0.4, \\ -2.6+2\xi, & 0.4 < \xi \le 1. \end{cases}$$

This problem is discretized by finite differences on a regular mesh with mesh size $h = 1/(N-1)$, where $N = n_y$:

$$\begin{aligned} y_\ell &:= y(\ell h), \quad \ell = 0,\dots,N, \\ u_\ell &:= u(\ell h), \quad \ell = 0,\dots,N, \\ -y''(\ell h) &\approx \frac{1}{h^2}(-y_{\ell-1}+2y_\ell-y_{\ell+1}), \quad \ell = 1,\dots,N-1, \\ \int_0^1 (y(\xi)-\bar{y}(\xi))^2 d\xi &\approx h\sum_{\ell=1}^{N-1}(y_\ell-\bar{y}(\ell h))^2, \\ \int_0^1 u(\xi)^2 d\xi &\approx h\sum_{\ell=1}^{N-1} u(\xi)^2. \end{aligned}$$

For the sake of simplicity, we omit values at 0 and 1 so that our vectors of unknowns are

$$y = (y_1,\dots,y_{N-1})^\top, \qquad u = (u_1,\dots,u_{N-1})^\top.$$

The discretized problem is now of the form (QP) with

$$H_{yy} = hI, \quad H_{yu} = H_{uy}^\top = 0, \quad H_{uu} = \nu h I, \qquad C_u = I, \qquad f_y = -\bar{y}, \quad f_u = 0, \quad c = 0,$$

where I is the identity in $\mathbb{R}^{N-1}$ and

$$C_y = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}.$$

In order to perform numerical convergence tests with varying approximations A to C_y, we construct these approximations by Jacobi steps; i.e., for $D := \operatorname{diag}(C_y)$ we define

$$A_0^{-1} := D^{-1}, \tag{4.34}$$
$$A_1^{-1} := D^{-1}(I + (I - C_y D^{-1})), \tag{4.35}$$
$$A_2^{-1} := D^{-1}(I + (I + (I - C_y D^{-1}))(I - C_y D^{-1})), \tag{4.36}$$
$$\vdots \quad \vdots$$

and therefore

$$A_i^{-1} = A_0^{-1}((A_0 - C_y)A_{i-1}^{-1} + I).$$

Thus

$$\begin{aligned} 1 > \rho(I - A_1^{-1}C_y) > \rho(I - A_2^{-1}C_y) = \rho(I - A_1^{-1}C_y)^2 > \rho(I - A_3^{-1}C_y) \\ = \rho(I - A_1^{-1}C_y)^3 > \cdots . \end{aligned}$$

In the case of A_0, we can give the Schur complement analytically:

$$S_{A_0} = H_{uu} + C_u^\top A_0^{-\top} H_{yy} A_0^{-1} C_u = \left(\nu h + \frac{h^5}{4}\right) I. \tag{4.37}$$

In all other cases, the formulas become more complicated. We treat B analogously: we choose $B_0 := H_{uu}$ and B_j^{-1} as the approximation to S_A^{-1} after j Richardson iterations with H_{uu}. That means

$$B_0 := H_{uu} = \mu h I, \tag{4.38}$$
$$B_j^{-1} := B_0^{-1}(I - C_u^\top A_i^{-\top} H_{yy} A_i^{-1} C_u B_{j-1}^{-1}). \tag{4.39}$$

Additionally, we investigate the cases $B = S_A$ and $B = S$ from (4.28).

We investigate for the setting $N = 101$ ($h = \frac{1}{100}$) and $\nu = 0.001$ the convergence performance. We perform GMRES iterations until the residual is below 10^{-6}, where we start at an iteration vector constant to zero. The problem solution is plotted in Figure 4.1.

Table 4.1 summarizes the results. In column i, we denote how many Jacobi iterations are performed for obtaining A_i^{-1} in (4.34)–(4.36). Column j gives analogous information for B_j^{-1}. The notation $j = S_A$ means that we choose $B = S_A$, and $j = S$ means that we choose $B = S$. Furthermore, ρ_A denotes the spectral radius of the matrix $I - A_i^{-1}C_y$. Analogously, ρ_S denotes the spectral radius of the iteration matrix of the Schur complement; cf. (4.39). In the last column "# it" refers to the number of GMRES iterations either without limitations on the history (GMRES(∞)) or with a history of only 30 vectors

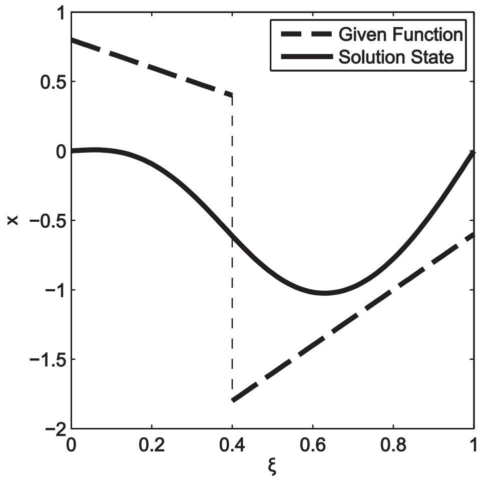

Figure 4.1. *Solution (state, solid) and desired function $\bar{y}$ (dashed) to be tracked. This figure first appeared in K. Ito, K. Kunisch, V. Schulz, and I. Gherman, Approximate nullspace iterations for KKT systems, SIAM J. Matrix Anal. Appl.,* 31(4) (2010), 1835–1847.

Table 4.1. *Convergence results for* $N = 101$ *and* $\nu = 0.001$.

				GMRES(∞)	GMRES(30)
i	j	ρ_A	ρ_S	# inner It.	# outer It.
—	—	—	—	201	—
0	0	0.9995	0.0000	181	—
0	1	0.9995	0.0000	181	—
0	S_A	0.9995	0.0000	181	—
0	S	0.9995	0.9113	182	—
10	0	0.9946	0.0003	56	7
10	1	0.9946	0.0000	56	7
10	S_A	0.9946	0.0000	56	7
10	S	0.9946	0.9112	57	8
20	0	0.9897	0.0011	41	3
20	1	0.9897	0.0000	41	3
5	S_A	0.9897	0.0000	41	3
5	S	0.9897	0.9112	42	4
100	0	0.9514	0.0243	19	1
100	1	0.9514	0.0006	19	1
100	S_A	0.9514	0.0000	19	1
100	S	0.9514	0.9091	19	1
1000	0	0.6101	0.1560	6	1
1000	S_A	0.6101	0.0000	7	1
1000	S	0.6101	0.7728	7	1

(GMRES(30)). A line in the last column means that convergence could not be achieved within 100 outer iterations of GMRES. The first line gives the result of the GMRES method without any preconditioning.

We observe that the GMRES iterations profits very much from a better approximation of the system matrix and also a slight advantage of the choice $B = S_A$.

4.5 SQP Methods and Variants

The idea of SQP for optimization problems of the type

$$\min f(y,u), \tag{4.40}$$

$$c(y,u) = 0, \tag{4.41}$$

$$h(y,u) \geq 0 \tag{4.42}$$

is to compute the solution in an iterative fashion, where a linear-quadratic subproblem has to be solved in each iteration. Since linear-quadratic problems are much better understood than general nonlinear problems, this approach is very attractive. SQP methods can be applied in very general settings concerning the vector spaces involved—in the most general setup they are derived from set-valued Newton methods. The embedding of SQP methods within appropriate function spaces is of major importance if one wants to guarantee a discretization independent good performance of the SQP iterations. We do not want to go into those details and refer the reader interested in an up-to-date overview to the book [194]. Here, we focus on the finite-dimensional setting and will add further aspects later, if appropriate.

For ease of presentation, we drop the inequalities in problem (4.40)–(4.42) and lump together the variables as $z := (y,u)$ in order to investigate the generic problem

$$\min f(z), \tag{4.43}$$

$$c(z) = 0. \tag{4.44}$$

We nevertheless keep in mind that all or some of the equality constraints might later represent so-called active inequality constraints. Optimization theory provides the following necessary conditions in terms of the Lagrangian

$$\partial L(z,p)/\partial z = 0, \qquad \text{where } L(z,p) := f(z) + p^\top c(z). \tag{4.45}$$

The principle of SQP methods starts at Newton's method for the necessary conditions (4.45) together with the constraints (4.44). This method iterates over z and the adjoint variables p in the form $(z^{k+1}, p^{k+1}) = (z^k, p^k) + (\Delta z^{k+1}, \Delta p^{k+1})$, where the increments solve the linear system

$$\begin{bmatrix} H & c_z^\top \\ c_z & 0 \end{bmatrix} \begin{pmatrix} \Delta z \\ \Delta p \end{pmatrix} = \begin{pmatrix} -\nabla f^k - c_z(z^k)^\top p^k \\ -c(z^k) \end{pmatrix}. \tag{4.46}$$

Here H denotes the Hessian of the Lagrangian L with respect to z, i.e., $H = L_{zz}(z^k, p^k)$, and subscripts denote respective derivatives. This equation is equivalent to a linear-quadratic optimization problem—essentially in two different formulations. We see immediately that the necessary conditions for QP

$$\min \frac{1}{2}\Delta z^\top H \Delta z + \left(\nabla f^k + c_z(z^k)^\top p^k\right)^\top \Delta z, \tag{4.47}$$

$$c_z \Delta z + c(z^k) = 0 \tag{4.48}$$

are equivalent to (4.46), where Δp are the adjoint variables in QP (4.47), (4.48). The solution of a QP is a challenging task by itself, and, in particular, in the presence of inequality constraints, the adjoint variables are involved in the solution process. For instance, in active-set strategies, the sign of the adjoint variables indicates whether an inequality is active or not. However, in formulation (4.47), (4.48), the adjoint variables Δp have no meaning, since they converge to zero anyway.

If we write Δp in (4.46) in the form $\Delta p = p^{k+1} - p^k$ and subtract the term $c_z(z^k)^\top p^k$ from both sides of the first row in (4.46), we end up with equivalent formulation

$$\begin{bmatrix} H & c_z^\top \\ c_z & 0 \end{bmatrix} \begin{pmatrix} \Delta z \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f^k \\ -c(z^k) \end{pmatrix}. \tag{4.49}$$

The equivalent linear-quadratic program is now

$$\min 1/2 \Delta z^\top H \Delta z + \nabla f^{k\top} \Delta z, \tag{4.50}$$

$$c_z \Delta z + c(z^k) = 0, \tag{4.51}$$

with adjoint variable p^{k+1}. If we assume that the algorithm converges to the solution of the optimization problem, then p^{k+1} converges to the adjoint variables of the nonlinear problem and can also be used in order to decide about the activity of inequality constraints.

Because the derivative generation needed to establish the matrices H and c_z often turns out to be prohibitively expensive, one often uses approximations instead, i.e., $G :\approx H$ and $A :\approx c_z$ (of course, these approximations may also change from iteration to iteration). This substitution will deteriorate the convergence behavior of the resulting SQP method. However, in view of (4.46) the fixed points of the iteration are not changed as long as the resulting matrix $\left[\begin{smallmatrix} G & A^\top \\ A & 0 \end{smallmatrix}\right]$ on the left-hand side is nonsingular. If the approximations G and A are sufficiently accurate, one may expect at least linear local convergence of the iteration method. In the implementation, one will terminate the iteration as soon as $(\Delta z^{k+1}, \Delta p^{k+1})$ is sufficiently close to zero in a suitable norm. If an estimate of the convergence rate is known, an a priori estimate for the distance to the limit point can be given.

In order to arrive at a generalization of this approach again to inequality constrained problems, we first reformulate the system of equations

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{pmatrix} \Delta z \\ \Delta p \end{pmatrix} = \begin{pmatrix} -\nabla f^k - c_z(z^k)^\top p^k \\ -c(z^k) \end{pmatrix} \tag{4.52}$$

in the form of an equivalent linear-quadratic problem. Rewriting again $\Delta p = p^{k+1} - p^k$ we obtain the formulation

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{pmatrix} \Delta z \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f^k + (A - c_z(z^k))^\top p^k \\ -c(z^k) \end{pmatrix}.$$

If G is symmetric and positive definite on the nullspace of A, this is equivalent to the linear-quadratic problem

$$\min 1/2\Delta z^\top G \Delta z + \left(\nabla f^k - (A - c_z(z^k))^\top p^k\right)^\top \Delta z, \tag{4.53}$$

$$A\Delta z + c(z^k) = 0, \tag{4.54}$$

where the adjoint variable for (4.54) is p^{k+1}. In this way, e.g., accurate evaluations of the matrices c_z or H can be avoided. Nevertheless, the matrix c_z appears in the objective of the QP, however, only in the form of a matrix-vector product with p^k, which can be efficiently realized, e.g., in the adjoint mode of automatic differentiation [154].

This formulation is generalizable to inequality constraints (cf. [40, 41]), e.g., of the form $c(z) \geq 0$ in (4.44), which leads to linear-quadratic subproblems of the form

$$\min 1/2\Delta z^\top G \Delta z + \left(\nabla f^k - (A - c_z(z^k))^\top p^k\right)^\top \Delta z, \tag{4.55}$$

$$A\Delta z + c(z^k) \geq 0. \tag{4.56}$$

The adjoint variables of (4.56) converge to the adjoint variables of the inequality constraints at the solution. Therefore, they provide a proper decision criterion within an active-set strategy. This formulation of SQP methods allowing for approximations of derivatives also forms the basis for the real-time investigations in [113].

By usage of the rather general formulation (4.55), (4.56), we can discuss several variants of SQP methods. We identify essentially two different cases.

Case 1: $A = c_z$ and H is approximated by G

This is the standard case around which the vast majority of research on SQP methods revolves. The interested reader should consult [265] for an in-depth discussion. Here, we highlight three major representatives for numerical methods:

Projected gradient method: $G = I$

In the unconstrained case, this method amounts to $\Delta z = -\tau \nabla f^k$ with a steplength parameter τ. In the constrained case, the step direction is chosen as the gradient projected to the feasible directions. Linear convergence speed towards the solution can be expected from this method.

Quasi-Newton method: G is constructed from updates

The change in the gradients from iteration to iteration gives an approximation to second order information in the step direction. This idea is used in update techniques. The most prominent is the BFGS-update constructed from the differences $q = \nabla_z L(z^{k+1}, p^{k+1}) - \nabla_z L(z^k, p^{k+1})$ and $s = z^{k+1} - z^k$ as [265]

$$G^{k+1} = G^k + \frac{qq^\top}{q^\top s} - \frac{G^k s (G^k s)^\top}{s^\top G^k s}.$$

It is essential for this method that the approximation G^k stay positive definite all over the iterations, although this can be expected only for H projected to the nullspace of the constraints. One can cope with this problem, e.g., by a modified update rule (Powell). Alternatively, trust region approaches are often employed for guaranteeing global convergence, which can tolerate lack of positive definiteness of G^{k+1}. The typical convergence speed for this SQP variant is superlinear.

Gauss–Newton methods for nonlinear least squares
If the objective $f(z)$ has a special structure like $f(z) = F(z)^\top F(z)$ for vector-valued mapping F, it is convenient to use the approximation $G = J^\top J$, where J is the Jacobian of F. This approximation ignores second-derivative terms of the Hessian of the Lagrangian, which is the reason for the convenience of this method. However, in particular in parameter estimation problems with a small final residual, the second-order terms present in $H - G$ are rather small such that the resulting method possesses a rather small linear convergence rate. This method is called the Gauss–Newton method in the unconstrained case and the generalized Gauss–Newton method [39] in the constrained case.

Reduced SQP
Reduced SQP methods focus on the Hessian of the Lagrangian projected to the nullspace of the linearized constraints. If this nullspace is the range space of a linear mapping T, then reduced SQP methods use the approximation $G = TBT^\top$, where B is an approximation to the reduced Hessian such that $B \approx T^\top HT$. If T spans the nullspace of all active constraints, we know that $T^\top HT$ is positive definite in the neighborhood of the solution if the optimization guarantees the existence of a locally unique solution. Thus, quasi-Newton update techniques as discussed above can be profitably applied for the generation of B. Furthermore, $T^\top HT$ inherits nonlinear least squares structures if they are present. If T spans a larger space than the kernel of all active constraints, the resulting method is called a partially reduced SQP method [310] and convergence results for quasi-Newton approximations depend on similarly unrealistic positive definiteness assumptions as in the case of full-space SQP methods. Partially reduced SQP methods are of particular advantage in the presence of inequality constraints. Reduced SQP methods are the basis for one-shot methods as discussed in the next section.

Case 2: Both c_z and H are approximated by A (resp., G)
If also c_z is approximated in (4.55), (4.56), the resulting methods can be called “approximate” methods, e.g., approximate reduced SQP methods. The convergence properties are not known in detail so far. The literature cited in the section on preconditioning ensures that the resulting approximate SQP variants are algorithms, which converge to the solution of the nonlinear optimization problem, provided the distance $c_z - A$ is small enough in some norm. However, the assumptions posed for convergence proofs typically are much too restrictive and those methods often do a good job also if these restrictive assumptions are not satisfied. Thus, there is still significant room for research.

The theoretical situation is better if the approximation quality of c_z is improved in each step of the SQP method in the fashion of an inexact Newton method. Then superlinear or even quadratic convergence properties can be regained, but with a cost significantly higher than in the approximate case. Most inexact Newton approaches are based on the investigations in [111]. Inexact (reduced) SQP methods based on trust region globalization have been extensively investigated in [187].

4.6 Reduced SQP and One-Shot Methods

The term “one-shot method” is used for solution methods for optimization problems, which solve the optimization problem during the solution of the state equation. It was coined by Ta’asan, Kuruvila, and Salas in [332] with a focus on multigrid methods. Various other names for the same algorithmic paradigm are in usage: boundary value problem approach, simultaneous design and analysis, piggy-back iterations. Thus the state equation and all other restrictions are never feasible during the optimization iterations besides at the optimal solution. From this point of view, already SQP methods are one-shot methods. However, usually the term one-shot methods is used for methods where not even linear subproblems are feasible during the iterations

We discuss again the separability framework, where problems of the type (2.1)–(2.3) are considered. First, assume the case without inequality constraints

$$\min f(y,u), \tag{4.57}$$
$$c(y,u) = 0. \tag{4.58}$$

The essential feature is the nonsingularity of c_y, which means that we can look at the problem as an unconstrained problem of the form

$$\min_u f(y(u),u). \tag{4.59}$$

When applied to the necessary optimality condition $\nabla_u f(y(u),u) = 0$, Newton's method, or its variants, yields good local convergence properties. Every iteration consists of two steps:

(1) Solve $B\Delta u = -\nabla_u f(y(u^k),u^k) = 0$, where $B \approx \nabla_u^2 f(y(u^k),u^k)$.

(2) Update $u^{k+1} = u^k + \tau\,\Delta u$, where τ is an appropriate step length.

The approximation B of the Hessian of f is often performed by quasi-Newton update formulas as discussed, e.g., in [265]. Step (1) of this algorithm involves two costly operations which, however, can be performed in a highly modular way. First $y(u^k)$ has to be computed, which means basically a full nonlinear solution of the flow problem abbreviated by (4.58). Furthermore, the corresponding gradient, $\nabla_u f(y(u^k),u^k)$, has to be determined. One of the most efficient methods for this purpose is the adjoint method, which means the solution of the linear adjoint problem, since for $y^k = y(u^k)$ one obtains

$$\nabla_u f(y(u^k),u^k) = f_u(y^k,u^k)^\top + c_u(y^k,u^k)^\top p,$$

where p solves the adjoint problem

$$c_y(y^k,u^k)^\top p = -f_y(y^k,u^k)^\top.$$

Assuming that (4.58) is solved for y by Newton's method, one might wonder whether it may be enough to perform only one Newton step per optimization iteration in the algorithm above. This results in the following algorithmic steps:

(1) Solve $\begin{bmatrix} 0 & 0 & c_y^\top \\ 0 & B & c_u^\top \\ c_y & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ p \end{pmatrix} = \begin{pmatrix} -f_y^\top \\ -f_u^\top \\ -c \end{pmatrix}$.

(2) Update $(y^{k+1},u^{k+1}) = (y^k,u^k) + \tau\,(\Delta y^{k+1},\Delta u^{k+1})$.

This algorithm is called a reduced SQP algorithm. The local convergence can be again of quadratic, superlinear, or linear type [228, 310], depending on how well B approximates the so-called reduced Hessian of the Lagrangian

$$B \approx L_{uu} - L_{uy}c_y^{-1}c_u - (L_{uy}c_y^{-1}c_u)^\top + c_u^\top c_y^{-\top} L_{yy} c_y^{-1} c_u.$$

The vector p produced in each step of the reduced SQP algorithm converges to the adjoint variable vector of problem (4.57), (4.58) at the solution. This iteration again can be written in the form of a Newton-type method as

$$\begin{bmatrix} 0 & 0 & c_y^\top \\ 0 & B & c_u^\top \\ c_y & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix} = \begin{pmatrix} -L_y^\top \\ -L_u^\top \\ -c \end{pmatrix}, \quad \begin{pmatrix} y^{k+1} \\ u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ u^k \\ p^k \end{pmatrix} + \tau \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix}. \tag{4.60}$$

This iteration can be generalized to inexact linear solves with an approximate matrix $A \approx c_y$ such that the approximate reduced SQP iteration reads as

$$\begin{bmatrix} 0 & 0 & A^\top \\ 0 & B & c_u^\top \\ A & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix} = \begin{pmatrix} -L_y^\top \\ -L_u^\top \\ -c \end{pmatrix}, \quad \begin{pmatrix} y^{k+1} \\ u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ u^k \\ p^k \end{pmatrix} + \tau \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix}. \tag{4.61}$$

It is shown in [211] that in this case the use of an approximation of the consistent reduced Hessian, i.e.,

$$B \approx L_{uu} - L_{uy}A^{-1}c_u - (L_{uy}A^{-1}c_u)^\top + c_u^\top A^{-\top} L_{yy} A^{-1} c_u,$$

is recommended. Let us compare this with the SQP formulation of (4.46) in the separability framework

$$\begin{bmatrix} L_{yy} & L_{yu} & c_y^\top \\ L_{uy} & L_{uu} & c_u^\top \\ c_y & c_u & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix} = \begin{pmatrix} -L_y^\top \\ -L_u^\top \\ -c \end{pmatrix}, \quad \begin{pmatrix} y^{k+1} \\ u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ u^k \\ p^k \end{pmatrix} + \tau \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix}. \tag{4.62}$$

Because of the triangular structure of the system matrix in (4.60) or (4.61), the reduced SQP formulation is much more modular than the full SQP formulation (4.62). This is the reason, why the matrices in (4.60) or (4.61) are used as preconditioners in large-scale linear-quadratic optimal control problems [26] or as preconditioners in Lagrange–Newton–Krylov methods as discussed in [37, 38]. Indeed, one observes for the (iteration) matrix

$$M = I - \begin{bmatrix} 0 & 0 & c_y^\top \\ 0 & B & c_u^\top \\ c_y & c_u & 0 \end{bmatrix}^{-1} \begin{bmatrix} L_{yy} & L_{yu} & c_y^\top \\ L_{uy} & L_{uu} & c_u^\top \\ c_y & c_u & 0 \end{bmatrix}$$

the fact that $M \neq 0$ in general, but $M^3 = 0$, which is the basis of the convergence considerations in [211].

Now, let us discuss one-shot methods in the context of optimization with PDEs as in, e.g., [183, 182]. We have

$$\min J(y,u) \qquad \text{(objective)}, \tag{4.63}$$

$$c(y,u) = 0 \qquad \text{(PDE model)}, \tag{4.64}$$

$$h(y,u) \geq 0 \qquad \text{(scalar constraints)}, \tag{4.65}$$

where y collects, e.g., the state variables of the model equation model and u is a finite-dimensional vector parameterizing the degrees of freedom for optimization. Often engineering knowledge tells us that the scalar constraint will be active at the solution. Therefore, we can formulate the constraint right from the beginning in the form of an equality constraint. In this context, a full SQP approach as in (4.62) reads as

$$\begin{bmatrix} L_{yy} & L_{yu} & h_y^\top & c_y^\top \\ L_{uy} & L_{uu} & h_u^\top & c_u^\top \\ h_y & h_u & 0 & 0 \\ c_y & c_u & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \mu \\ \Delta p \end{pmatrix} = \begin{pmatrix} -L_y^\top \\ -L_u^\top \\ -h \\ -c \end{pmatrix}, \quad \begin{pmatrix} y^{k+1} \\ u^{k+1} \\ \mu^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ u^k \\ \mu^k \\ p^k \end{pmatrix} + \tau \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \mu \\ \Delta p \end{pmatrix}. \tag{4.66}$$

This approach is not implementable in general, because one usually starts out with a PDE solver for $c(y,u) = 0$ and seeks a modular coupling with an optimization approach, which does not necessitate a change to the whole code structure, as would be the case with formulation (4.62). A modular but nevertheless efficient alternative is an approximate reduced SQP approach as in (4.61), which is adapted to the case of the additional lift (or pitching) constraint, as established in [143],

$$\begin{bmatrix} 0 & 0 & 0 & A^\top \\ 0 & B & \gamma & c_u^\top \\ 0 & \gamma^\top & 0 & 0 \\ A & c_u & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \mu \\ \Delta p \end{pmatrix} = \begin{pmatrix} -L_y^\top \\ -L_u^\top \\ -h \\ -c \end{pmatrix}, \quad \begin{pmatrix} y^{k+1} \\ u^{k+1} \\ \mu^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ u^k \\ \mu^k \\ p^k \end{pmatrix} + \tau \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta \mu \\ \Delta p \end{pmatrix}, \tag{4.67}$$

where

$$\gamma = h_u^\top + c_u^\top \alpha \text{ such that } A^\top \alpha = -h_y^\top .$$

An algorithmic version of this modular formulation of a generic one-shot scheme is given by the following algorithm.

ALGORITHM 4.11. One-shot scheme.

(1) generate p^k by performing N iterations of an adjoint solver with right-hand side $J_y^\top(y^k, u^k)$ starting in p^k

(2) generate α^k by performing N iterations of an adjoint solver with right-hand side $h_y^\top(y^k, u^k)$ starting in α^k

(3) compute approximate reduced gradients

$$g = J_u^\top + c_u^\top p^{k+1}, \quad \gamma = h_u^\top + c_u^\top \alpha^{k+1}$$

(4) generate B_{k+1} as an approximation of the consistent reduced Hessian

(5) solve the QP

$$\begin{bmatrix} B & \gamma \\ \gamma^\top & 0 \end{bmatrix} \begin{pmatrix} \Delta u \\ \mu^{k+1} \end{pmatrix} = \begin{pmatrix} -g \\ -h \end{pmatrix}$$

(6) update $u^{k+1} = u^k + \Delta u$

(7) adjust the computational mesh, if necessary

(8) generate y^{k+1} by performing N iterations of the forward state solver starting from an interpolation of y^k at the new mesh.

This highly modular algorithmic approach is not an exact transcription of (4.67), but is shown in [143] to be asymptotically equivalent and to converge to the same solution. The overall algorithmic effort for this algorithm is typically in the range of factor 7 to 10 compared to a forward stationary simulation. Usually, step (6) has to be enhanced by a step relaxation which aims at keeping the forward residual below a desirable level. In Section 7.2.5, we discuss an application of this technique with a particular choice for the Hessian approximation B.

Another one-shot approach favored in [137] is strongly correlated with automatic differentiation. In contrast to the block-Gauss–Seidel-type iteration (4.61), this one-shot method applied to problem (4.57), (4.58) takes the form of a block Jacobi iteration

$$\begin{bmatrix} 0 & 0 & A^\top \\ 0 & B & 0 \\ A & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix} = \begin{pmatrix} -L_y^\top \\ -L_u^\top \\ -c \end{pmatrix}, \quad \begin{pmatrix} y^{k+1} \\ u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} y^k \\ u^k \\ p^k \end{pmatrix} + \tau \begin{pmatrix} \Delta y \\ \Delta u \\ \Delta p \end{pmatrix}. \tag{4.68}$$

A step damping is performed on the basis of a primal and dual augmented Lagrangian. Furthermore, it is generalized to additional scalar restrictions by the usage of a penalty multiplier method as in [138].