Math. Prog. Comp. (2016) 8:435–459
DOI 10.1007/s12532-016-0101-2

FULL LENGTH PAPER

An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal Hessian matrix

Dennis Janka[1] · **Christian Kirches**[1] · **Sebastian Sager**[2] · **Andreas Wächter**[3]

Received: 23 January 2015 / Accepted: 15 February 2016 / Published online: 29 February 2016

Abstract We present a quasi-Newton sequential quadratic programming (SQP) algorithm for nonlinear programs in which the Hessian of the Lagrangian function is block-diagonal. Problems with this characteristic frequently arise in the context of optimal control; for example, when a direct multiple shooting parametrization is used. In this article, we describe an implementation of a filter line-search SQP method that computes search directions using an active-set quadratic programming (QP) solver. To take advantage of the block-diagonal structure of the Hessian matrix, each block is approximated separately by quasi-Newton updates. For nonconvex instances, that arise, for example, in optimum experimental design control problems, these blocks are often found to be indefinite. In that case, the block-BFGS quasi-Newton update can lead to poor convergence. The novel aspect in this work is the use of SR1 updates in place of BFGS approximations whenever possible. The resulting indefinite QPs necessitate an inertia control mechanism within the sparse Schur-complement factorization

✉ Dennis Janka
dennis.janka@iwr.uni-heidelberg.de

Christian Kirches
christian.kirches@iwr.uni-heidelberg.de

Sebastian Sager
sager@ovgu.de

Andreas Wächter
andreas.waechter@northwestern.edu

1 Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany

2 Institute for Mathematical Optimization, Otto-von-Guericke-University Magdeburg, 39106 Magdeburg, Germany

3 Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208, USA

that is carried out by the active-set QP solver. This permits an adaptive selection of the Hessian approximation that guarantees sufficient progress towards a stationary point of the problem. Numerical results demonstrate that the proposed approach reduces the number of SQP iterations and CPU time required for the solution of a set of optimal control problems.

Keywords Quasi-Newton · Sequential quadratic programming · Direct methods for optimal control · Optimum experimental design

Mathematics Subject Classification 49M37 · 90C20 · 90C30 · 90C53 · 90C55

1 Introduction

We propose a sequential quadratic programming (SQP) algorithm for the solution of nonlinear programs (NLPs) in which the Hessian of the Lagrangian function is block-diagonal. This structure arises frequently in the context of optimization problems that are constrained by systems of ordinary differential equations (ODEs), for example, during the optimal control [8] or optimum experimental design [28,31,32,39,41] of dynamic processes. When direct [44] or all-at-once methods [14], such as direct multiple shooting [7,33,37], are applied, the block-diagonal structure is a consequence of the discretization of the control trajectory and the parametrization of the state trajectory in time. In this article, we are concerned with quasi-Newton updates that replace the individual blocks of the block-diagonal Hessian matrix when their exact computation is too costly. Related approaches have been proposed in [7,12,26,27], for example. In slight abuse of mathematical rigour, we will call these matrices "Hessian approximation", although the updated matrices do usually not converge in any sense to the exact second derivative.

Because these blocks are often not positive definite in general nonlinear nonconvex optimal control problems, methods based on the block-wise BFGS update can lead to poor convergence (see Sect. 1.1). Whenever the BFGS formula would yield an indefinite matrix, it is common to modify the update formula so that it results in a damped approximation that lies between the current positive definite approximation and the one produced by the unmodified BFGS formula. In this way, positive definiteness is maintained [38]. In the following, we always assume that damping of the BFGS update takes place if necessary.

In optimal control problems, we typically find many differential state variables that are restricted by continuity conditions. This leaves only a small number of degrees of freedom represented by those control variables that are not at their bounds. Hence, in each iteration of a nonlinear programming method, the null space of the active constraints will be small compared to the number of variables. It then becomes likely that the constraint range space contains negative eigenvalues of the Hessian matrix, so that BFGS updates must be modified to maintain positive definiteness. This often turns out to be a bad choice because it leads to slow convergence and an increased number of costly function and derivative evaluations.

In contrast to previous SQP methods, the proposed algorithm addresses this shortcoming by employing the indefinite, block-wise Symmetric Rank One (SR1) update in place of the block-wise BFGS approximation whenever this is possible. The SR1 update has long been known for its good theoretical properties and has been used successfully in unconstrained optimization [9,11,30]. It is usually employed in connection with a trust region [10]. Instead, our method determines the inertia of the Hessian matrix in the appropriate subspace in order to decide whether the search direction generated by the block-SR1 approximation provides sufficient descent for a filter line search [45]. If this is not the case, the search direction is recomputed with the block-BFGS approximation as a fallback. This procedure requires the solution of nonconvex quadratic subproblems. Such problems are NP-hard in general [36], and we content ourselves with computing first order critical points. Our implementation uses the active-set QP solver qpOASES [19,20] that was modified to monitor the inertia of the submatrix during the active-set updates. This approach is based on a sparse matrix factorization [18], a Schur complement decomposition [5,23–25], and a flipping bounds strategy [20]. Numerical results demonstrate that the new algorithm requires fewer SQP iterations and less CPU time than the pure block-BFGS method for a number of optimal control problems.

1.1 Motivating nonlinear programming example

To motivate the key contribution of this article, we consider the simple unconstrained convex minimization problem

$$\min_{x_1 \in \mathbb{R}} \quad \tfrac{1}{2} x_1^2.$$

This example illustrates that the introduction of additional variables and constraints can lead to negative curvature that impairs the performance of the block-BFGS update. A more complicated example is discussed in Sect. 5.2. After adding a second variable x_2 along with the constraint $x_1 = x_2$, we obtain the equivalent problem

$$\min_{x_1, x_2 \in \mathbb{R}} \quad x_1^2 - \tfrac{1}{2} x_2^2 \tag{1a}$$

$$\text{s.t.} \quad x_1 - x_2 = 0. \tag{1b}$$

Because the constraint that couples x_1 and x_2 is linear, the Hessian of the Lagrangian function (6) of NLP (1),

$$\nabla^2_{xx} \mathcal{L}(x, \lambda) = \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix},$$

is block diagonal. Furthermore, it has a negative eigenvalue in the range space of (1b). To study the convergence behavior of different quasi-Newton updates, we choose identity as the initial Hessian approximation and declare optimality when

$$\frac{\|\nabla \mathcal{L}(x, \lambda)\|_\infty}{1 + \|\lambda\|_\infty} < \epsilon \quad \text{and} \quad \|c(x)\|_\infty < \epsilon \tag{2}$$

is satisfied with $\epsilon = 10^{-12}$. In [29, Chapter 5.3] it is shown for this example that a full-step method with block-wise SR1 updates or full-space SR1 or BFGS updates recovers the solution after at most three iterations but converges only linearly with convergence rate $\frac{1}{2}$ if damped block-wise BFGS updates are used. At the solution, we have $\lambda = 0$. While this may suggest some kind of degeneracy in the example, the algorithms behave essentially identical if the right-hand side of the constraint is changed to, e.g., one, which yields a nonzero multiplier. Note that the linear independence constraint qualification (LICQ) holds at any point.

The theoretical convergence behavior is confirmed by our implementation: Using the starting point $(x_1, x_2) = (10, 10)$, we observe that the standard full-space SR1 update formula recovers the exact Hessian $\nabla^2_{xx}\mathcal{L}(x, \lambda)$ after two SQP iterations. Hence, the optimal solution of problem (1) is found after three iterations. The block-SR1 approximation scheme, which exploits the knowledge of linear separability to approximate the nonzero diagonal entries of $\nabla^2_{xx}\mathcal{L}(x, \lambda)$ separately, recovers the exact Hessian already after the first iteration and finds the optimal solution after two iterations.

This situation is entirely different when damped BFGS updates are used. After two iterations, the standard BFGS approximation yields the positive definite matrix

$$\begin{pmatrix} 4.5 & -2.5 \\ -2.5 & 1.5 \end{pmatrix}. \tag{3}$$

Damping is not necessary to maintain positive definiteness. The entries of (3) do not approximate those of the exact Hessian. In particular, the block structure is lost. Nevertheless, the termination criterion (2) is satisfied after three SQP iterations. Finally, the block-BFGS scheme produces approximations of the form

$$\begin{pmatrix} 2 & 0 \\ 0 & \varepsilon \end{pmatrix}, \quad \varepsilon > 0,$$

where ε converges to zero. This method requires 43 SQP iterations until the termination criterion is satisfied. This behavior is similar to that of the steepest descent method, which needs 45 iterations for the problem at hand.

Although Example (1) may be a special situation from the viewpoint of general nonlinear optimization, it is very relevant from the viewpoint of numerical optimal control. There, block-diagonal structured Hessians arise in the context of direct multiple shooting [7].

In this paper, we propose an algorithm that maintains the advantages of block-wise updates and avoids the poor performance of the damped BFGS approximation. This is done by using SR1 updates whenever the resulting search direction has descent properties that are required for the global convergence of the method.

1.2 Contributions and outline

The remainder of the article is laid out as follows. In Sect. 2 we present an active-set SQP method with a filter line-search globalization strategy. The method makes it possible to work with indefinite Hessian matrices or their approximations, and it provides

a suitable framework for our numerical investigations. In Sect. 3 we discuss several strategies for the indefinite block-wise approximation of the Hessian. We present criteria that decide when to use the block-SR1 or the block-BFGS approximation with the goal of promoting accelerated local convergence. In Sect. 4 we describe recent extensions to the active-set parametric QP solver qpOASES that concern the efficient solution of sparse and block-structured problems. We also discuss a new inertia control strategy in qpOASES that permits the identification of points that satisfy second-order necessary optimality conditions in active-set form (e.g. [24], Result 2.2) of an indefinite QP. We study the behavior of the proposed algorithm in detail in Sect. 5, using an optimum experimental design control example. Section 6 evaluates the merits of the proposed SQP algorithm on a number of representative application problems from optimum experimental design and optimal control. Finally, Sect. 7 concludes the article.

1.3 Notation

Throughout the article, subscripts, as in x_i, indicate the elements of a column vector. Bracketed superscript indices, as in $x^{[k]}$, denote quantities belonging to an iteration. The avoid cumbersome notation, we distinguish the type of iteration by the index letter: SQP iterations k, limited-memory iterations j, and working set iterations ν of the active-set QP solver.

2 A filter line-search SQP method with indefinite hessian approximations

In this section, we present an SQP method that is based on the filter line-search globalization proposed in [45]. It solves nonlinear programs of the form

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{4a}$$

$$\text{s.t.} \quad c(x) = 0, \tag{4b}$$

$$\ell \le x \le u. \tag{4c}$$

Here, the objective function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and the constraint function $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ with $m < n$ are assumed to be twice continuously differentiable. The vectors $\ell \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are the lower and upper bounds on the variables x. For future reference, the Lagrange multipliers of (4b) are denoted by $\lambda \in \mathbb{R}^n$.

Starting with an initial guess $x^{[0]}$, we compute iterates $x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}$ in SQP iteration $k \ge 0$. The search direction $d^{[k]}$ is given as the solution of the quadratic program QP($H^{[k]}$)

$$\min_{d\in\mathbb{R}^n} \quad \tfrac{1}{2} d^T H^{[k]} d + {g^{[k]}}^T d \tag{5a}$$

$$\text{s.t.} \quad A^{[k]} d + c^{[k]} = 0, \tag{5b}$$

$$\ell \le x^{[k]} + d \le u. \tag{5c}$$

In QP (5), the matrix $A^{[k]}$ is the constraint Jacobian $\nabla_x c(x^{[k]})^T$, and the matrix $H^{[k]}$ is an approximation of the Hessian of the Lagrangian function,

$$\nabla^2_{xx}\mathcal{L}(x^{[k]}, \lambda^{[k]}) = \nabla^2 f(x^{[k]}) + \sum_{j=1}^{m} \lambda_j^{[k]} \nabla^2 c(x^{[k]}). \tag{6}$$

The step size $\alpha^{[k]}$ is determined by the filter line search described in [45]. A step is accepted if it sufficiently reduces either the objective value $f(x)$ or the constraint violation $\|c(x)\|_\infty$. A feasibility restoration phase may be necessary to generate iterates that reduce the constraint violation when the step size $\alpha^{[k]}$ in direction $d^{[k]}$ falls below a certain threshold.

Under the assumption that the approximations $H^{[k]}$ are uniformly positive definite on a certain subspace, it is guaranteed that a solution of (5) exists and that the resulting search direction has descent properties that are required for the filter line search. In order to formulate the positive-definiteness condition, let $d^{[k]}$ be the critical point found by the QP solver and let us define $\mathcal{S}^{[k]}$ as the set of bounds that are active at *both* $x^{[k]}$ *and* $x^{[k]} + d^{[k]}$:

$$\mathcal{S}^{[k]} := \left\{ 1 \le i \le n \mid (x_i^{[k]} = \ell_i \text{ or } x_i^{[k]} = u_i) \text{ and } d_i^{[k]} = 0 \right\}. \tag{7}$$

Furthermore, let $\mathcal{A}^{[k]}$ be the matrix whose rows consist of the gradients of the equality constraints and of the bounds that are active at *both* $x^{[k]}$ *and* $x^{[k]} + d^{[k]}$:

$$\mathcal{A}^{[k]} := \left[(A^{[k]})^T \; e_{j_1} \cdots e_{j_{l_k}} \right]^T. \tag{8}$$

Here, $\mathcal{S}^{[k]} = \{j_1, \ldots, j_{l_k}\}$, and e_i is the i-th coordinate vector. With these definitions in place, we can formally state the assumption.

Assumption 1 There exists a constant $\lambda_1 > 0$, so that for each k the smallest eigenvalue of $(Z^{[k]})^T H^{[k]} Z^{[k]}$ is at least λ_1, where $Z^{[k]}$ is a matrix whose columns form an orthonormal basis of the null space of $\mathcal{A}^{[k]}$.

In the rest of the paper, we will refer to $(Z^{[k]})^T H^{[k]} Z^{[k]}$ as the "reduced Hessian $\bar{H}^{[k]}$ (corresponding to $\mathcal{A}^{[k]}$)".

Global convergence can be shown if Assumption 1 (named (G3*) in [45]) and further assumptions, such as boundedness of the function and their derivatives and well-definedness of the restoration phase, hold. In a practical implementation, it is difficult to guarantee *uniform* positive definiteness for the entire sequence $\{\bar{H}^{[k]}\}$ as required by Assumption 1, because this would require a very large amount of time for the computation of the smallest eigenvalue. Instead, our SQP method guarantees positive definiteness of $\bar{H}^{[k]}$ in every individual iteration k only, which does not preclude that zero may be a limit point of the smallest eigenvalues of $\bar{H}^{[k]}$. Consequently, in theory, it is possible that Assumption 1 may not be satisfied and the method is no longer guaranteed to converge; it might generate arbitrarily large steps so that the method reverts to the restoration phase too often and the limit points of the iterates are feasible

but not necessarily stationary points. However, this situation is very unlikely in most practical situations, and heuristic approaches similar to the one described here have been successfully use in other optimization codes, including `Ipopt` [46] and `Loqo` [43].

Algorithm 1 gives the description of the overall SQP method. It is identical with Algorithm I in [45], with some details in step 2. As motivated in Sect. 1.1, the method maintains two block quasi-Newton approximations of the Hessian: (a) a block-SR1 approximation that might be indefinite but permits the approximation of negative curvature in the constraint range space ; and (b) a block-BFGS approximation that is guaranteed to be positive definite. The algorithm first attempts to use the block-SR1 update, and if the corresponding reduced Hessian is positive definite, the resulting step is taken as the search direction. Otherwise, the method recomputes the search direction with the positive definite block-BFGS approximation.

The following section describes in detail how the Hessian approximations are generated.

Algorithm 1 Filter Line Search SQP Algorithm with Indefinite Hessian Matrices

Given: Starting point $x^{[0]}$. Set $k = 0$.

1. *Check convergence.* Stop if $x^{[k]}$ is a stationary point of the NLP.
2. *Compute search direction.*
   (a) Construct the block Hessian approximations $H_{\mathrm{SR1}}^{[k]}$ and $H_{\mathrm{BFGS}}^{[k]}$.
   (b) Solve QP($H_{\mathrm{SR1}}^{[k]}$) to obtain a primal-dual critical point $(d_{\mathrm{SR1}}^{[k]}, \tilde{\lambda}_{\mathrm{SR1}}^{[k]})$.
   (c) If QP($H_{\mathrm{SR1}}^{[k]}$) is infeasible, go to 5.
   (d) If the reduced Hessian $\bar{H}_{\mathrm{SR1}}^{[k]}$ corresponding to $\mathcal{A}^{[k]}$ is positive definite, set $(d^{[k]}, \tilde{\lambda}^{[k]}) = (d_{\mathrm{SR1}}^{[k]}, \tilde{\lambda}_{\mathrm{SR1}}^{[k]})$, set $H^{[k]} = H_{\mathrm{SR1}}^{[k]}$, and go to 3.
   (e) Solve QP($H_{\mathrm{BFGS}}^{[k]}$) to obtain a primal-dual critical point $(d_{\mathrm{BFGS}}^{[k]}, \tilde{\lambda}_{\mathrm{BFGS}}^{[k]})$.
   (f) Set $(d^{[k]}, \tilde{\lambda}^{[k]}) = (d_{\mathrm{BFGS}}^{[k]}, \tilde{\lambda}_{\mathrm{BFGS}}^{[k]})$ and set $H^{[k]} = H_{\mathrm{BFGS}}^{[k]}$.
3. *Backtracking line search.* Try to find a step length $\alpha^{[k]}$ by the filter line search as described in [45]. If the trial step size becomes too small, go to 5.
4. *Next iteration.* Set

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}, \qquad \lambda^{[k+1]} = (1 - \alpha^{[k]})\lambda^{[k]} + \alpha^{[k]}\tilde{\lambda}^{[k]},$$

   $k \leftarrow k + 1$, and go to 1.
5. *Feasibility restoration phase.* If possible, compute $x^{[k+1]}$ that is accepted by the filter as described in [45] and go to 1. Otherwise, stop and declare the problem as first-order locally infeasible.

3 Partitioned quasi-Newton updates

We maintain the block-diagonal structure of the true Hessian of the Lagrangian function with a partitioned quasi-Newton update. Following the approach suggested in [7], we approximate each block separately by a suitable update formula. This can be seen as a special case of partitioned quasi-Newton updates proposed in [26,27] and leads to a high-rank update in each SQP iteration. The updates apply the full-space formulae to the appropriate subvectors of

$$s^{[k]} := x^{[k+1]} - x^{[k]}, \qquad y^{[k]} := \nabla_x \mathcal{L}(x^{[k+1]}, \lambda^{[k+1]}) - \nabla_x \mathcal{L}(x^{[k]}, \lambda^{[k+1]}).$$

Likewise, the scaling and damping procedures described below are carried out for each block independently.

In the following, we refrain from introducing an additional index to indicate a block. Rather, the quantities $s^{[k]}$, $y^{[k]}$, $H_{(\cdot)}^{[k]}$, $\theta^{[k]}$, and $\gamma_{(\cdot)}^{[k]}$ refer to any one of the diagonal blocks in the Hessian for the remainder of this section. Formulae should hence be read as being applied to each block individually, and to all blocks at the same time.

3.1 Limited-memory storage and build-up

The active-set QP solver `qpOASES` [20] used in our implementation requires that the elements of the Hessian matrix are provided explicitly. As a consequence, we compute the dense quasi-Newton matrices for each block. Nevertheless, we chose the limited-memory approximation over the full-memory update because we noticed in our experiments that using full-memory BFGS updates often leads to an increase in the number of SQP iterations required, see Sect. 6.3.

For the limited-memory update, we store the M most recent vectors

$$\{s^{[k-1]}, \ldots, s^{[k-M]}\}, \ \{y^{[k-1]}, \ldots, y^{[k-M]}\},$$

where $M = \min(\tilde{M}, k)$ and $\tilde{M}$ is a fixed number. Starting with an initial approximation H_{ini}, we explicitly construct the blocks of the matrix $H^{[k]}$ by forming the sum of the M most recent updates. This approach is efficient because the blocks are small, and the computational cost for constructing the dense matrices is negligible compared to the derivative computation and QP solution.

In the first SQP iteration, we set $H_{\text{ini}} = I$. In a later iteration k, the initial approximation H_{ini} is chosen as the identity matrix multiplied by the scaling factor

$$\gamma_{\text{ini}}^{[k]} = \frac{{y^{[k]}}^T s^{[k]}}{{s^{[k]}}^T s^{[k]}}$$

due to Oren and Luenberger [35]. We explored alternative initializations, but the experiments reported in [29] showed that this strategy was most efficient in our setting.

3.2 SR1 update

To avoid cluttered notation, we will omit the iteration index k for the remainder of this section. The index j refers to the steps in the limited-memory update in a given iteration k. The SR1 update uses the standard formula

$$H_{\text{SR1}}^{[j+1]} = H_{\text{SR1}}^{[j]} + \frac{(y^{[j]} - H_{\text{SR1}}^{[j]} s^{[j]})(y^{[j]} - H_{\text{SR1}}^{[j]} s^{[j]})^T}{(y^{[j]} - H_{\text{SR1}}^{[j]} s^{[j]})^T s^{[j]}}, \quad j = k - M, \ldots, k - 1.$$

Following the rule in [34], we guard against denominators that are close to zero and skip the update whenever

$$\left|(y^{[j]} - H_{\mathrm{SR1}}^{[j]} s^{[j]})^T s^{[j]}\right| < \epsilon_0 \cdot \|(y^{[j]} - H_{\mathrm{SR1}}^{[j]} s^{[j]})\|_2 \cdot \|s^{[j]}\|_2.$$

In our implementation, we choose $\epsilon_0 = 10^{-8}$.

3.3 Damped BFGS update

We use Powell's damping strategy [38] that makes it possible to perform the BFGS update even in a direction of negative curvature. For $j = k - M, \ldots, k - 1$, damping parameters $\theta^{[j]}$ are computed from

$$\theta^{[j]} = \begin{cases} \dfrac{0.8 s^{[j]T} H_{\mathrm{BFGS}}^{[j]} s^{[j]}}{s^{[j]T} H_{\mathrm{BFGS}}^{[j]} s^{[j]} - s^{[j]T} y^{[j]}}, & \text{if } s^{[j]T} y^{[j]} < 0.2 s^{[j]T} H_{\mathrm{BFGS}}^{[j]} s^{[j]}, \\ 1, & \text{else.} \end{cases}$$

With this, we define $\overline{y}^{[j]}$ by

$$\overline{y}^{[j]} = \theta^{[j]} y^{[j]} + (1 - \theta^{[j]}) H_{\mathrm{BFGS}}^{[j]} s^{[j]},$$

and use $\overline{y}^{[j]}$ in place of $y^{[j]}$ to compute the damped update:

$$H_{\mathrm{BFGS}}^{[j+1]} = H_{\mathrm{BFGS}}^{[j]} - \frac{H_{\mathrm{BFGS}}^{[j]} s^{[j]} s^{[j]T} H_{\mathrm{BFGS}}^{[j]}}{s^{[j]T} H_{\mathrm{BFGS}}^{[j]} s^{[j]}} + \frac{\overline{y}^{[j]} \overline{y}^{[j]T}}{\overline{y}^{[j]T} s^{[j]}}, \quad j = k - M, \ldots, k - 1.$$

We store the unmodified vectors $y^{[j]}$ for the limited-memory update, because the damping parameters are different for each SQP iteration. Note that $\theta^{[j]} = 1$ gives the unmodified BFGS update.

An additional safeguard is required because the update described above is applied block-wise, i.e., $s^{[j]}$ is in fact only a subvector of the full-space step. This implies that $s^{[j]}$ and hence the denominators $s^{[j]T} H^{[j]} s^{[j]}$ and $\overline{y}^{[j]T} s^{[j]}$ can be zero even though the full-space step is not. We skip the update for the current block whenever $s^{[j]T} H^{[j]} s^{[j]} < \epsilon_1$ or $\overline{y}^{[j]T} s^{[j]} < \epsilon_1$. In our implementation, we choose $\epsilon_1 = 10^{-14}$.

3.4 Selective sizing of the limited-memory updates

We observed in our experiments that the BFGS update sometimes produces approximations with very large eigenvalues. As a remedy, we consider the option to scale the approximation $H_{\mathrm{BFGS}}^{[j]}$ before each (inner) iteration in the limited-memory update. This is based on the *selective sizing* strategy by Contreras and Tapia [13], which aims to prevent the accumulation of large eigenvalues in the quasi-Newton approximation

that may lead to unnecessarily small steps. We use the *centered Oren–Luenberger sizing factor*

$$\gamma_{\mathrm{COL}}^{[j]} = \frac{1}{2} \frac{\bar{y}^{[j-1]^T} s^{[j-1]} / s^{[j-1]^T} s^{[j-1]} + \bar{y}^{[j]^T} s^{[j]} / s^{[j]^T} s^{[j]}}{s^{[j-1]^T} H^{[j]} s^{[j-1]} / s^{[j-1]^T} s^{[j-1]} + s^{[j]^T} H^{[j]} s^{[j]} / s^{[j]^T} s^{[j]}}$$

as follows: Whenever $0 < \epsilon_2 < \gamma_{\mathrm{COL}}^{[j]} < 1$ for a given $\epsilon_2 \in (0, 1)$, we reset $H^{[j]} \leftarrow \gamma_{\mathrm{COL}}^{[j]} H^{[j]}$ before the computation of the update $H^{[j+1]}$. In our implementation, we choose $\epsilon_2 = 10^{-1}$. For SR1 updates, this option made no significant difference and has not been used in our experiments.

4 Solution of sparse nonconvex quadratic programs with qpOASES

Step 2 of Algorithm 1 requires the solution of QP($H^{[k]}$) (with $H^{[k]} = H_{\mathrm{SR1}}^{[k]}$ and potentially with $H^{[k]} = H_{\mathrm{BFGS}}^{[k]}$). The QP solver qpOASES in our implementation works with the parametric QP

$$\min_d \tfrac{1}{2} d^T H^{[k]} d + g^T(\tau) d \tag{9a}$$

$$\text{s.t. } A^{[k]} d + c(\tau) = 0, \tag{9b}$$

$$\ell - x^{[k]} \le d \le u - x^{[k]}. \tag{9c}$$

Here, the gradient vector $g(\tau)$ and the constraint vector $c(\tau)$ are affine-linear functions parameterized by $\tau \in [0, 1]$, given by

$$g(\tau) = \tilde{g}^{[k-1]}(1 - \tau) + g^{[k]}\tau, \qquad c(\tau) = \tilde{c}^{[k-1]}(1 - \tau) + c^{[k]}\tau.$$

The parametric data in $\tau = 0$ is initialized with

$$\tilde{g}^{[k-1]} = g^{[k-1]} + (H^{[k,l]} - H^{[k-1]}) d^{[k-1]} - (A^{[k]} - A^{[k-1]})^T \tilde{\lambda}^{[k-1]},$$
$$\tilde{c}^{[k-1]} = c^{[k-1]} + (A^{[k]} - A^{[k-1]}) d^{[k-1]}.$$

This makes it possible to warm-start the solver in $\tau = 0$ from the known optimal solution $(d^{[k-1]}, \tilde{\lambda}^{[k-1]})$ of the previous SQP iteration. The solution of (9) in $\tau = 1$ is the solution of QP($H^{[k]}$) sought for. We refer to [20] and the references therein for the details of the parametric active-set strategy implemented in qpOASES.

In the following, we focus on two new features that were implemented in the context of this article. The first exploits the sparsity of $H^{[k]}$ and $A^{[k]}$ using a sparse Schur complement approach. The second feature is an inertia control mechanism that determines whether the reduced Hessian $\bar{H}^{[k]}$ corresponding to $\mathcal{A}^{[k]}$ defined in (8) is positive definite.

4.1 Schur-complement approach

Every active-set iteration ν in the parametric active-set QP method requires the solution of linear systems involving the KKT matrix

$$K^{[\nu]} := \begin{bmatrix} H^{[\nu]} & A^{[\nu]^T} \\ A^{[\nu]} & 0 \end{bmatrix}. \tag{11}$$

Here, $H^{[\nu]}$ is the matrix composed from the rows and columns of $H^{[k]}$ with indices in the working set of free variables, $\mathcal{F}^{[\nu]}$. The matrix $A^{[\nu]}$ consists of the columns of $A^{[k]}$ with indices in $\mathcal{F}^{[\nu]}$. The working set $\mathcal{F}^{[\nu]}$ is chosen such that $A^{[\nu]}$ has full row rank and such that $\bar{H}^{[\nu]}$ is regular. Furthermore, the complement of $\mathcal{F}^{[\nu]}$, denoted by $(\mathcal{F}^{[\nu]})^{\complement}$, contains only fixed variables active at their bounds; i.e., $(\mathcal{F}^{[\nu]})^{\complement} \subseteq \{i \mid d_i^{[\nu]} = \ell_i - x_i^{[k]} \text{ or } d_i^{[\nu]} = u_i - x_i^{[k]}\}$.

We follow the approach described by [5,23–25] and compute a symmetric indefinite LBL^T-factorization of (11) for the initial working set $\mathcal{F}^{[0]}$. In our implementation, this is accomplished by the sparse multifrontal solver `MA57` [18] with approximate minimum degree ordering computed by `MC47` [1]. Before the first QP iteration, pivot information from `MA57` is used to identify a subset of rows with full rank for a given working set guess. Then, the exchange logic of an active-set method (here `qpOASES`) maintains this property over the course of all active-set QP iterations. Details can be found in [20].

The working set $\mathcal{F}^{[\nu]}$ changes over iterations ν affect the rows and columns present in (11), and this must be reflected in the factorization. As shown in [23], we border the matrix (11) by appropriately chosen matrices $M^{[\nu]}$ and $N^{[\nu]}$:

$$K^{[0]} = \begin{bmatrix} H^{[0]} & A^{[0]^T} \\ A^{[0]} & 0 \end{bmatrix}, \qquad K^{[\nu]} := \begin{bmatrix} K^{[0]} & M^{[\nu]} \\ M^{[\nu]^T} & N^{[\nu]} \end{bmatrix}, \quad \nu = 1, 2, \ldots.$$

The dimension of the matrices $M^{[\nu]}$ and $N^{[\nu]}$ grows or shrinks by one in each iteration. The key insight is that solutions with $K^{[\nu]}$ only require solutions with $K^{[0]}$ and with the dense Schur complement $S^{[\nu]} = N^{[\nu]} - M^{[\nu]^T} K^{[0]^{-1}} M^{[\nu]}$. The LBL^T-factors for $K^{[0]}$ are computed only once at the beginning, and we compute and update a dense QR factorization of $S^{[\nu]}$.

4.2 Refactorization and restoring linear independence

Because the cost for maintaining a dense QR factorization of the Schur complement $S^{[\nu]}$ grows with the size of $S^{[\nu]}$, a new LBL^T-factorization is computed when the size of $S^{[\nu]}$ exceeds a given threshold `nSmax`. A refactorization is also triggered when an estimate of the condition number of $S^{[\nu]}$ is larger than a given threshold `condMax`. In our implementation, the estimate is computed by the `LAPACK` routine `DTRCON` which is applied to the triangular factor R of the QR decomposition. We found that the values `nSmax` $= 100$ and `condMax` $= 10^{14}$ work well in practice.

It is possible that the LBL^T-factorization cannot be performed because the matrix $K^{[\nu]}$ is singular. This can occur because of numerical error in the linear-independence test due to finite precision. Similarly, already the initial KKT matrix $K^{[0]}$ may be singular because the constraint Jacobian $A^{[k]}$ and Hessian approximation $H^{[k]}$ are different at the new iterate.

In both cases, we make use of a feature of the symmetric indefinite solver `MA57` and obtain the list of zero pivot indices. This enables us to remove linearly dependent rows and columns from $K^{[\nu]}$ by manipulating the working set $\mathcal{F}^{[\nu]}$ appropriately. According to this list, we either (1) remove a variable from the working set $\mathcal{F}^{[\nu]}$ if the zero pivot is found in the first block diagonal of (11), or (2) add a dummy slack variable $0 \leq s \leq 0$ to an equality constraint and add it to the working set $\mathcal{F}^{[\nu]}$ if the zero pivot corresponds to the second, all-zero block diagonal in (11).

4.3 Monitoring the inertia of the KKT matrix

Throughout the working set iterations, we ensure that the reduced Hessian $\bar{H}^{[\nu]}$ corresponding to $A^{[\nu]}$ is always positive definite. To this purpose, we monitor the inertia $\text{In}(K^{[\nu]}) = (n_+, n_-, n_0)$ of the full KKT matrix $K^{[\nu]}$ (11), where n_+, n_-, and n_0 are the number of positive, negative, and zero eigenvalues of $K^{[\nu]}$. Using the techniques described next, we then make sure that n_+ equals the size of the first block diagonal in (11), and n_- equals the size of the second block diagonal in (11). This implies positive-definiteness of $\bar{H}^{[\nu]}$ [34].

We start with an initial KKT matrix $K^{[0]}$ with correct inertia. The case of incorrect inertia is addressed in Sect. 4.4. We then follow the approach presented in [25] that makes use of the fact that the inertia of $K^{[\nu]}$ is correct if and only if

$$\text{In}(S^{[\nu]}) = (\sigma_+, \sigma_-, 0), \tag{12}$$

where σ_+ is the number of variables added and σ_- is the number of variable removed from the set $\mathcal{F}^{[\nu]}$ since the most recent LBL^T-factorization of the KKT matrix. The inertia of $K^{[\nu]}$ can thus be determined from the inertia of $S^{[\nu]}$, and it can be tracked efficiently by observing the sign of the determinant of $S^{[\nu]}$ in every modification of $S^{[\nu]}$.

If a variable is removed from the working set, the reduced Hessian shrinks by one dimension and remains positive definite. Now consider a variable x_i that is added to the working set and that subsequently the KKT matrix has no longer the desired inertia. By the properties of the working set, x_i must be at one of its bounds (9c), say $x_i(\tau) = l_i(\tau) - x_i^{[k]}$. `qpOASES` treats this situation using the *flipping bounds* strategy [20]. Here, x_i is removed again from the working set and fixed to its bound on the opposite side, $x_i(\tau) = u_i(\tau) - x_i^{[k]}$. In the parametric active-set strategy, however, the value of x_i is not actually changed. Instead, the parametric upper bound is redefined, $u_i(\tau) := l_i(\tau)$, and primal feasibility is preserved. We point out that this procedure does not necessarily guarantee convergence. An alternative strategy, due to Fletcher [21], reduces the quadratic objective by moving along a direction of negative curvature and can be shown to terminate finitely in a non-parametric QP solver. (At this point

it is not clear to us whether a parametric variant of Fletcher's approach also has finite convergence).

4.4 Warm-start and early termination

For each subproblem in Step 2 of Algorithm 1, qpOASES is warm-started using the primal-dual solution and working set of the previous SQP iteration. If the resulting KKT matrix is singular, the working set is reduced according to the procedure described in Sect. 4.2.

In step 2b, it is important to determine whether the reduced Hessian $\bar{H}_{\mathrm{SR1}}^{[k]}$ corresponding to the optimal solution $d_{\mathrm{SR1}}^{[k]}$ is positive definite. If the initial working set yields a KKT matrix with correct inertia, the parametric active-set algorithm is carried out to find a critical point of QP($H_{\mathrm{SR1}}^{[k]}$). Afterwards, we factorize the KKT matrix corresponding to the subset $\mathcal{S}^{[k]}$ of the working set and obtain its inertia from the linear solver.

In case the inertia of the KKT matrix $K^{[0]}$ is not correct for the initial working set $\mathcal{F}^{[0]}$, we can conclude without solving the QP that the reduced Hessian $\bar{H}_{\mathrm{SR1}}^{[k]}$ corresponding to the solution $d_{\mathrm{SR1}}^{[k]}$ cannot be positive definite because the set $\mathcal{S}^{[k]}$ in (7) is a subset of $(\mathcal{F}^{[0]})^{\mathsf{C}}$. The solution of QP($H_{\mathrm{SR1}}^{[k]}$) is then terminated immediately and the method proceeds to Step 2e.

5 Numerical case study: optimum experimental design

Our work is motivated by the solution of nonconvex optimal control problems. As a basic example, we consider a dynamic process $y(\cdot) \in \mathbb{R}^{n_y}$ that is modeled by a system of ordinary differential equations on a fixed time horizon $[t_0, t_f] \subset \mathbb{R}$ and depends on a set of time-independent parameters $p \in \mathbb{R}^{n_p}$. The process can be controlled by a control function $u(\cdot) \in \mathbb{R}^{n_u}$ which is constrained by bounds, $b_l \le u(\cdot) \le b_u$. The optimal control problem is then written as

$$\min_{y(\cdot),u(\cdot),p} \quad \phi(y(t_f)) \tag{13a}$$

$$\text{s.t.} \quad \dot{y}(t) = f(t, y(t), p, u(t)) \qquad t \in [t_0, t_f], \tag{13b}$$

$$y(t_0) = y_0, \tag{13c}$$

$$b_l \le u(t) \le b_u \qquad t \in [t_0, t_f]. \tag{13d}$$

A particularly interesting subclass are optimum experimental design (OED) control problems. OED problems occur as subproblems in the process of model validation, where unknown model parameters $p \in \mathbb{R}^{n_p}$ are to be identified, see Fig. 1.

We consider an ODE system of the form

$$\dot{y}(t) = f(t, y(t), p, u(t)), \quad y(t_0) = y_0.$$

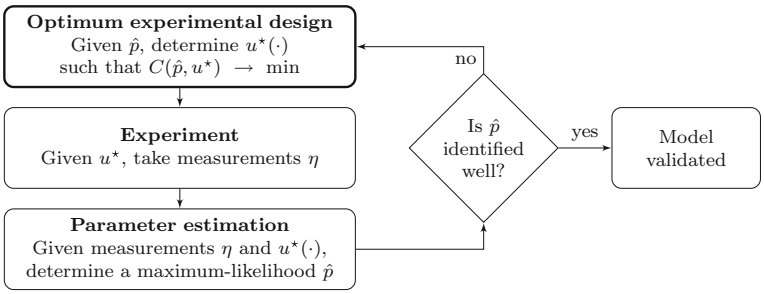

Fig. 1 Model validation flowchart. The focus of this article is on the numerical solution of the optimum experimental design problem

The control $u(\cdot)$ represents experimental conditions. In OED one is interested in finding the control $u(\cdot)$ that minimizes the uncertainty of a given estimate $\hat{p}$. The estimate is obtained, e.g., from previously taken measurements. We quantify the uncertainty by the trace of the covariance matrix of the parameter estimates, also called the A-criterion, a common objective in experimental design [39]. Suppose we can take observations $\eta_i \in \mathbb{R}^{n_y}$ of the differential state $y(t_i)$ at times $t_i \in [t_0, t_f]$, $0 \leq i \leq M$. The matrices

$$C(\hat{p}) = H(\hat{p})^{-1}, \quad H(\hat{p}) := \sum_{i=0}^{M} y_p(t_i)^T y_p(t_i) \tag{14}$$

are the covariance matrix C and Fisher information matrix H that describe the uncertainty of the maximum-likelihood estimate $\hat{p}$ that can be obtained from the data η [6]. Herein, $y_p(\cdot) := \frac{\mathrm{d}y(\cdot)}{\mathrm{d}p}$ is the solution of the variational ODE system evaluated at $\hat{p}$

$$\dot{y_p}(t) = f_y(t, y(t), \hat{p}, u(t)) \cdot y_p(t) + f_p(t, y(t), \hat{p}, u(t)), \quad y_p(t_0) = 0,$$

on $[t_0, t_f]$, and $f_y := \frac{\partial f}{\partial y}$ and $f_p := \frac{\partial f}{\partial p}$ denote the partial derivatives of f. We assume that the OED problem is well posed, that is, all components of p are identifiable. This implies regularity of the Fisher information matrix $H(\hat{p})$. If this assumption is violated in practice, prior information about p can be incorporated in the formulation by adding a term ϵI to the right-hand side of Eq. (14).

Notably, both matrices are independent of the data η itself. It is this observation that enables us to determine $u(\cdot)$ optimally and predict the covariance $C(\hat{p})$ for a given estimate $\hat{p}$ *before* we obtain the data. For nonlinear systems, the choice of $u(\cdot)$ not only affects $C(\hat{p})$ but $\hat{p}$ itself. In practice, however, we expect that the parameter estimates converge to their true values after few iterations of the loop in Fig. 1. Hence, $C(\hat{p})$ is a good estimate for the true covariance matrix.

In summary, the OED problem can be cast as the following optimal control problem, see [28,41]:

$$\min_{y(\cdot), y_p(\cdot), u(\cdot), C} \quad \operatorname{tr}(C) \tag{15a}$$

$$\begin{aligned}
\text{s.t.} \qquad \dot{y}(t) &= f(t, y(t), \hat{p}, u(t)) && t \in [t_0, t_\mathrm{f}], && \text{(15b)}\\
y(t_0) &= y_0, && && \text{(15c)}\\
\dot{y_\mathrm{p}}(t) &= f_\mathrm{y}(t, y(t), \hat{p}, u(t)) \cdot y_\mathrm{p}(t) \\
&\quad + f_\mathrm{p}(t, y(t), \hat{p}, u(t)) && t \in [t_0, t_\mathrm{f}], && \text{(15d)}\\
y_\mathrm{p}(t_0) &= 0, && && \text{(15e)}\\
b_\mathrm{l} \le u(t) &\le b_\mathrm{u} && t \in [t_0, t_\mathrm{f}], && \text{(15f)}\\
C &= \left(\sum\nolimits_{i=0}^{M} y_\mathrm{p}(t_i)^T y_\mathrm{p}(t_i) \right)^{-1}. && && \text{(15g)}
\end{aligned}$$

5.1 Direct multiple shooting

To solve this problem numerically, it is discretized in time and transformed into an NLP by a special variant of the direct multiple shooting method [7].

1. Choose a (possibly coarse) partition of the time horizon into $N > 1$ intervals, $t_0 = \tau_0 < \tau_1 < \cdots < \tau_N = t_\mathrm{f}$. To ease the notation, we will assume that the points τ_j coincide with the observation times t_i in (15g), hence $N = M$. In practice, however, it is often beneficial to choose separate grids.
2. Construct a piecewise discretization of the control $u(\cdot)$ on the intervals $[\tau_j, \tau_{j+1})$, represented by finite-dimensional control vectors q_j. For simplicity of exposition, we assume a piecewise constant discretization,

$$u(t) = q_j \in \mathbb{R}^{n_\mathrm{u}}, \quad t \in [\tau_j, \tau_{j+1}).$$

3. Introduce intermediate initial values $s_j \in \mathbb{R}^{n_\mathrm{y}}$ and $S_j \in \mathbb{R}^{n_\mathrm{y} \times n_\mathrm{p}}$ for $0 \le j \le N$. The initial value problem (15b)–(15e) is solved independently on each of the N intervals. Add continuity conditions to ensure equivalence to the original problem:

$$\begin{aligned}
y(\tau_{j+1};\ \tau_j, s_j, q_j) - s_{j+1} &= 0,\\
y_\mathrm{p}(\tau_{j+1};\ \tau_j, s_j, S_j, q_j) - S_{j+1} &= 0, \qquad j = 0, \ldots, N-1.
\end{aligned}$$

   Here, $y(\tau_{j+1};\ \tau_j, s_j, q_j)$ is the value of y at time τ_{j+1} obtained by integrating (15d) starting at τ_j with initial value s_j and control value q_j. The expression $y_\mathrm{p}(\tau_{j+1};\ \tau_j, s_j, S_j, q_j)$ is defined in a similar way.
4. Linear separability of the objective is achieved through a linearly coupled constraint for the inverse $H = C^{-1}(\hat{p})$ of the covariance matrix,

$$\sum_{j=1}^{N} y_\mathrm{p}(\tau_j)^T y_\mathrm{p}(\tau_j) - H = 0.$$

   This preserves the block-diagonal Hessian structure introduced by the multiple shooting discretization. Note that this would not be the case if the constraint was formulated in terms of C.

The resulting discretized and parameterized problem is an NLP in the variables

$$\begin{aligned} z_j &= (s_j, S_j, q_j) \in \mathbb{R}^{n_\mathrm{y}+n_\mathrm{y}n_\mathrm{p}+n_\mathrm{u}}, \quad 0 \le j \le N-1, \\ z_N &= (s_N, S_N) \in \mathbb{R}^{n_\mathrm{y}+n_\mathrm{y}n_\mathrm{p}}, \\ H &\in \mathbb{R}^{n_\mathrm{p}\times n_\mathrm{p}}. \end{aligned}$$

With $z := (z_0, \ldots, z_N)$, this NLP reads

$$\begin{aligned} \min_{z,H} \quad & \mathrm{tr}(H^{-1}) && && \text{(16a)} \\ \text{s.t.} \quad & 0 = y(\tau_{j+1}; s_j, q_j) - s_{j+1} && 0 \le j \le N-1, && \text{(16b)} \\ & 0 = y_0 - s_0, && && \text{(16c)} \\ & 0 = y_\mathrm{p}(\tau_{j+1}; s_j, S_j, q_j) - S_{j+1} && 0 \le j \le N-1, && \text{(16d)} \\ & 0 = S_0, && && \text{(16e)} \\ & b_\mathrm{l} \le q_j \le b_\mathrm{u} && 0 \le j \le N-1, && \text{(16f)} \\ & 0 = \sum\nolimits_{j=0}^{M} S_j^T S_j - H. && && \text{(16g)} \end{aligned}$$

Direct multiple shooting offers several advantages. Among others, it allows for the straightforward parallelization of state and derivative evaluation, the efficient treatment of DAE systems via a relaxed formulation, and the robust solution of the underlying problem with specialized ODE solvers. Furthermore, efficient linear algebra techniques may be employed for the solution of the NLP [33].

5.2 Numerical case study

We discuss a minimal, but prototypical experimental design example. It allows an analytical investigation, in contrast to the more complex problems in the next section. We consider the ODE system

$$\dot{y}(t) = p \cdot u(t), \quad y(t_0) = 0, \tag{17}$$

with one scalar parameter p and given estimate $\hat{p} = 2$ on the time horizon $[t_0, t_f] = [0, 1]$. Let the control $u(\cdot)$ be constrained by $0 \le u(t) \le 1$. The variational differential equation corresponding to (17) is

$$\dot{y}_p(t) = u(t), \quad y_p(t_0) = 0.$$

With a piecewise-constant discretization of $u(\cdot)$, the ODE solutions on the shooting intervals $[\tau_j, \tau_{j+1}]$, $0 \le j < N$ have the representations

$$\begin{aligned} y(\tau_{j+1};\ \tau_j, s_j, q_j) &= s_j + p q_j (\tau_{j+1} - \tau_j), && \text{(18a)} \\ y_\mathrm{p}(\tau_{j+1};\ \tau_j, s_j, S_j, q_j) &= S_j + q_j (\tau_{j+1} - \tau_j). && \text{(18b)} \end{aligned}$$

Independent of N, the global minimum of this problem is $\mathrm{tr}((H^*)^{-1}) = 4.5796 \cdot 10^{-2}$. It is attained at $q_j = 1, 0 \leq j < N$. This can be seen by applying a maximum principle [41]. Hence, the system reveals the most information about the parameter p if we excite it by the maximum amount possible.

If we insert equations (18) into the NLP (16), we note that the only nonlinear constraint is (16g); its Lagrange multiplier is denoted by $\lambda_{\mathrm{H}} \in \mathbb{R}^{n_{\mathrm{p}}}$. Linear separability with respect to the variables z_j, $j = 0, \ldots, N$, implies the following block-diagonal structure of the Hessian of the Lagrangian $\mathcal{L}(z, \lambda_{\mathrm{H}})$ of (16):

$$\nabla^2_{z,z}\mathcal{L}(z, \lambda_{\mathrm{H}}) = \mathrm{diag}\left(\nabla^2_{z_0,z_0}\mathcal{L}(z, \lambda_{\mathrm{H}}), \ldots, \nabla^2_{z_N,z_N}\mathcal{L}(z, \lambda_{\mathrm{H}}), \nabla^2_{H,H}\mathcal{L}(z, \lambda_{\mathrm{H}})\right),$$

$$\nabla^2_{z_j,z_j}\mathcal{L}(z, \lambda_{\mathrm{H}}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2\lambda_{\mathrm{H}} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \nabla^2_{H,H}\mathcal{L}(z, \lambda_{\mathrm{H}}) = 2H^{-3}. \tag{19}$$

While $2H^{-3}$ is positive close to the optimal solution, $2\lambda_{\mathrm{H}}$ is not. In our example, we have $\lambda^*_{\mathrm{H}} = -2.1 \cdot 10^{-3}$. Consequently, we expect that the SQP algorithm with block-wise updates performs better if the updates can reflect the negative curvature of the exact Hessian.

To examine this, we choose a grid of $N = 65$ equidistant points on $[0, 1]$. As initial guess, we set $q_j = 0.9$ and we initialize s_j and S_j such that all continuity conditions (18) are satisfied. The initial Hessian approximation is identity. With this setup, our SQP implementation finds the solution in 31 iterations with damped block-LBFGS updates. In all iterations, the LBFGS update is damped in 64 out of the 65 blocks to preserve positive definiteness of the approximation, or they are skipped entirely due to ill-conditioning as described in Sect. 3.3. In comparison, the block-LSR1/LBFGS SQP algorithm needs only 14 iterations to converge. In nine of these iteration, the block-LSR1 update is rejected by the inertia condition and a damped block-LBFGS approximation is employed instead.

At the solution, the exact Hessian (19) has 64 negative eigenvalues corresponding to the 64 intervals. If we choose a coarser multiple shooting grid for the state parameterization, but keep the control discretization grid fixed, we obtain a family of NLPs. In each instance, the number of control degrees of freedom is the same and the reduced Hessian at the solution has the same dimension. Table 1 shows the number of SQP iterations taken for these NLPs. We see that when block-SR1 updates are used, the number of SQP iterations decreases when the grid is refined. When block-BFGS updates are used this effect can also be observed, but the performance gain is smaller.

These results were carried out without the scaling heuristics. Table 1 also shows the number of iterations when we apply the block-LBFGS and block-LSR1 updates in connection with the scaling strategies presented in Sect. 3.4. Here, the number of iterations can be reduced significantly, especially for coarse grids. For the finest grid, where the nodes of the multiple shooting grid and the control grid coincide, the number of iterations for block-LBFGS is still considerably higher, which supports our use of block-LSR1 updates in these situations.

We remark that a full-space LBFGS update (that does not exploit the block-diagonal structure) with identity as initial approximation takes 55 iterations for this problem,

Table 1 Number of SQP iterations taken to solve example (16) to a KKT tolerance (Eq. 2) of 10^{-12}

N	No scaling		With scaling	
	b-BFGS	b-SR1 (rej.)	b-BFGS	b-SR1 (rej.)
2	55	47 (23)	9	23 (1)
3	52	44 (23)	9	22 (1)
5	48	38 (24)	10	14 (2)
9	54	30 (22)	10	11 (4)
17	46	24 (19)	12	11 (6)
33	36	18 (13)	15	9 (4)
65	31	14 (9)	28	10 (5)

N is the number of multiple shooting nodes. Iteration counts are reported for the block-LBFGS and block-LSR1 updates. For the latter, the number in parentheses is the number of iterations in which the block-LSR1 update is not accepted in Step 2 and the block-LBFGS update is used instead. The experiments were performed with and without the scaling heuristics. In the unscaled case, identity is taken as initial approximation. In the other case, the initial approximation for SR1 is scaled by γ_{ini} defined in Sect. 3.1. LBFGS updates are scaled in every limited-memory-update iteration using the selective sizing described in Sect. 3.4. The memory size is set to $\tilde{M} = 20$ as in all our numerical experiments

independent of N. When scaled with the selective sizing strategy, the number of iterations is reduced to 8. Results for the full-space BFGS update on more complicated examples are presented in Sect. 6.3.

6 Numerical results

In this section we examine the performance of our algorithm on a wider set of problems from optimal control and optimum experimental design.

6.1 Software implementation

The SQP algorithm presented in this paper is implemented in the C++ code `blockSQP`. It can be obtained from https://github.com/djanka2/blockSQP. It uses `qpOASES` release *3.2.0* available from https://projects.coin-or.org/qpOASES to solve the quadratic subproblems. Within `qpOASES`, `MA57` [18] is used to solve sparse linear systems involving the initial KKT matrix.

The objective and constraint evaluation is handled by the software package VPLAN [31]. It calls the variable step-size BDF method DAESOL [2,3] to evaluate the differential states in the continuity conditions and their derivatives. The required derivatives of the model right-hand side are supplied by ADIFOR [4].

All results were obtained on a workstation with 4 Intel Xeon CPUs (3.3 GHz) and 256 GB memory running CentOS Linux 7.0.1406. All C++ source code was compiled with `gcc` and all Fortran code with `gfortran` with the `-O3` option. For our experiments, only one core is used.

Table 2 Numbers n of variables and m of constraints of the NLPs in the test set. The number of multiple shooting discretization nodes is 64 for all problems

Problem	n	m
`oc-batchdist`	1092	1027
`oc-cops-catalyst-mixing`	257	193
`oc-cops-goddard`	322	258
`oc-cops-hangglider`	386	324
`oc-cops-hanging-chain`	257	195
`oc-cops-particle-steering`	450	388
`oc-fermenter`	900	708
`oc-lotka`	322	257
`oc-ocean`	322	194
`oc-williams-otto`	706	578
`oed-catalyst-mixing`	447	255
`oed-lotka`	383	383
`oed-urethane`	3331	3124

6.2 Test set and algorithmic parameters

Our set of test instances comprises the problems listed in Table 2. From the COPS 3.0 test set [17], we consider the optimal control problems 4 (hanging chain), 9 (particle steering), 10 (Goddard rocket), 11 (hang glider), and 14 (catalyst mixing). The catalyst mixing model can also be formulated as an optimum experimental design problem with the frequency factor of x_2 as uncertain parameter and both states as observables. The Lotka–Volterra optimal control and experimental design models are described in [42] and [40], respectively. The Williams-Otto semi-batch reactor is described in [22], and the batch distillation process in [15]. The Urethane reaction was first described as optimum experimental design problem in [31]. Both the fermenter and the ocean problem are found in the MUSCOD-II collection of optimal control problems [7,33].

All problems are discretized using the direct multiple shooting method outlined in Sect. 5 with $N = 64$ multiple shooting intervals. The relative integration tolerances in DAESOL are set between 10^{-7} and 10^{-9}, depending on the dynamical system. Table 2 shows the characteristics of the resulting NLPs.

All NLPs have a block-diagonal Hessian matrix. Block-wise quasi-Newton updates are computed as described in Sect. 3. The limited memory size is set to $\tilde{M} = 20$. The SQP method is terminated when the KKT error (2) is less than 10^{-5}.

6.3 Comparison of different update strategies

First, we evaluate the performance of our implementation of Algorithm 1 for the test set with different Hessian update strategies. Figure 2 compares the number of SQP iterations for six different strategies in the form of a performance profile [16]:

- block-wise BFGS, full memory (b-BFGS)
- block-wise BFGS, limited memory (b-LBFGS)

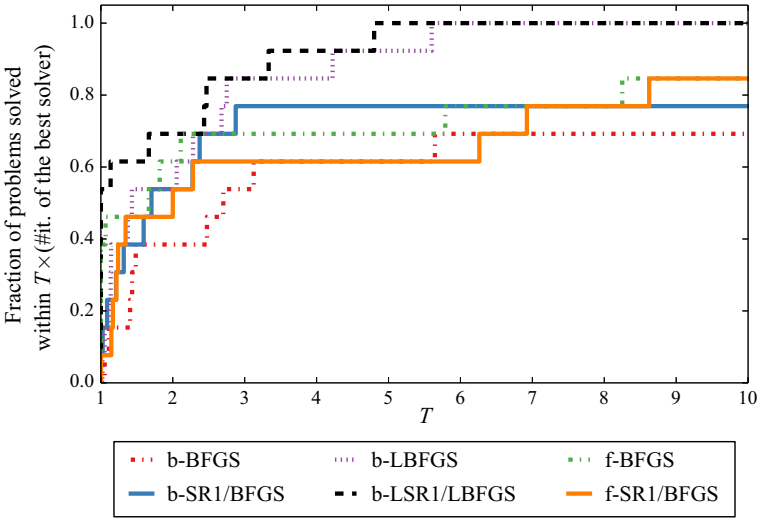

Fig. 2 Performance profile comparing the number of SQP iterations for six different update strategies

- block-wise SR1 with block-wise BFGS as fallback, full memory (b-SR1/BFGS)
- block-wise SR1 with block-wise BFGS as fallback, limited memory (b-LSR1/LBFGS)
- full-space BFGS, full memory (f-BFGS)
- full-space SR1 with BFGS as fallback, full memory (f-SR1/BFGS)

Some problems could not be solved by all algorithms due to errors in the line search or restoration phase

- `oc-cops-hangglider`: b-BFGS, b-SR1/BFGS, f-BFGS, f-SR1/BFGS
- `oc-batchdist`: b-BFGS, f-BFGS
- `oed-lotka`: b-BFGS, b-SR1/BFGS
- `oed-urethane`: f-SR1/BFGS
- `oc-fermenter`: b-SR1/BFGS
- `oc-cops-catalyst-mixing`: b-BFGS

We see that block-wise updates with limited memory perform better than their full-memory counterparts. We attribute this to the fact that "old" curvature information is retained by the full-memory update. Recall that the constraints involve the solution of dynamical systems with an error-controlled numerical integration routine that uses adaptively generated grids. If the integration grid changes from one SQP iteration to another, certain numerical noise is introduced in the secant information. We speculate that this can result in poor Hessian approximations that prevent fast progress of the algorithm.

Furthermore, we see that full-space BFGS updates sometimes find solutions in fewer SQP iterations than block-wise updates, see Table 3 for details. In some cases (`oc-cops-catalyst-mixing`, `oc-cops-hanging-chain`) where the algorithm with block-wise updates needs significantly more iterations, many updates are skipped because of ill-conditioning as described in Sect. 3.3. Note that full-space

Table 3 SQP iterations and run times for full-space BFGS and block-wise LSR1/LBFGS

Instance	f-BFGS		b-LSR1/LBFGS		
	SQP it	Time (s)	SQP it	Time (s)	Skip upd.(%)
oc-batchdist	73[a]	430.09[a]	*66*	*6.11*	0.4
oc-cops-catalyst-mixing	*45*	*1.45*	90	1.58	47.6
oc-cops-goddard	*37*	2.48	42	*0.48*	2.3
oc-cops-hangglider	8[a]	4.39[a]	*51*	*0.83*	0.7
oc-cops-hanging-chain	*62*	1.72	151	*1.04*	19.8
oc-cops-particle-steering	110	27.61	*19*	*0.5*	0.5
oc-fermenter	*32*	29.69	79	*4.87*	6.5
oc-lotka	38	3.05	*18*	*0.33*	2.1
oc-ocean	330	94.1	*40*	*1.68*	0.4
oc-williams-otto	*24*	30.07	40	*2.8*	11.5
oed-catalyst-mixing	49	3.05	*46*	*0.69*	33.2
oed-lotka	*25*	4.72	120	*2.9*	2.2
oed-urethane	180	13,947.56	*99*	*551.94*	1.2

For the latter, the percentage of updates that are skipped due to ill-conditioning is given. Smaller numbers are printed in italics

[a] Failures due to errors in the line search or restoration phase

updates are not competitive in terms of run times because block-sparsity in the Hessian cannot be exploited and much time is spent for solving the dense linear systems inside the QP solver.

In Table 4, we compare b-LSR1/LBFGS updates to b-LBFGS updates in more detail. We note that the b-LSR1/LBFGS method needs fewer SQP iterations for 12 out of 13 problems. Interestingly, SR1 updates are accepted relatively rarely by the inertia controlling strategy. Figure 3 shows in which particular iterations the SR1 updates are accepted. This is mostly the case during the last iterations, when the active set has settled and the reduced Hessian with exact derivatives, corresponding to the active constraints, is positive definite.

The KKT error threshold of 10^{-5} used to obtain these results may seem relatively loose. Tighter thresholds would even play in favor of the proposed approach as they require better numerical approximations of the ODE solution. These come at a higher relative cost per SQP iteration, which makes solving up to two relatively cheap QPs per iteration even more affordable.

Furthermore, we note that the savings in CPU time are not as significant as the reduction of SQP iterations. Sometimes, the b-LSR1/LBFGS method is even slower although fewer SQP iterations are needed. This is mainly due to the fact that many additional QPs need to be solved. Note, however, that the total number of QP iterations is often not much higher for the b-LSR1/LBFGS algorithm compared to the pure b-LBFGS case. The reason for this is that we terminate the QP solver early whenever the initial active set has the wrong inertia (see Sect. 4.4). Table 5 shows that this technique saves many QP iterations and is necessary to make the algorithm competitive with

Table 4 SQP iterations, QP iterations, and run times for b-LSR1/LBFGS and b-LBFGS

Instance	b-LSR1/LBFGS			b-LBFGS		
	SQP it	QP it	Time (s)	SQP it	QP it	Time (s)
oc-batchdist	*66* (7)	*622*	*13.83*	75	642	14.95
oc-cops-catalyst-mixing	*90* (2)	1415	*2.48*	114	*1250*	2.54
oc-cops-goddard	*42* (5)	*313*	*0.75*	76	491	1.36
oc-cops-hangglider	*51* (5)	417	*1.73*	73	*385*	2.31
oc-cops-hanging-chain	*151* (7)	*303*	*1.63*	166	363	1.63
oc-cops-particle-steering	*19* (2)	337	0.7	20	*321*	*0.65*
oc-fermenter	79 (2)	*1367*	6.5	*73*	1781	*5.94*
oc-lotka	*18* (4)	507	0.48	20	*355*	*0.41*
oc-ocean	*40* (4)	5167	*2.06*	55	*4919*	2.29
oc-williams-otto	*40* (5)	*1750*	*3.75*	66	1759	4.62
oed-catalyst-mixing	*46* (5)	650	*1.33*	47	*533*	1.58
oed-lotka	*120* (8)	*1738*	*4.92*	140	2025	5.53
oed-urethane	*99* (8)	*5669*	*549.33*	113	10028	600.84

Listed in parentheses is the number of SQP iterations in which the SR1 update was accepted by the inertia controlling strategy. The BFGS approximation was used in all other iterations. Smaller numbers are printed in italics

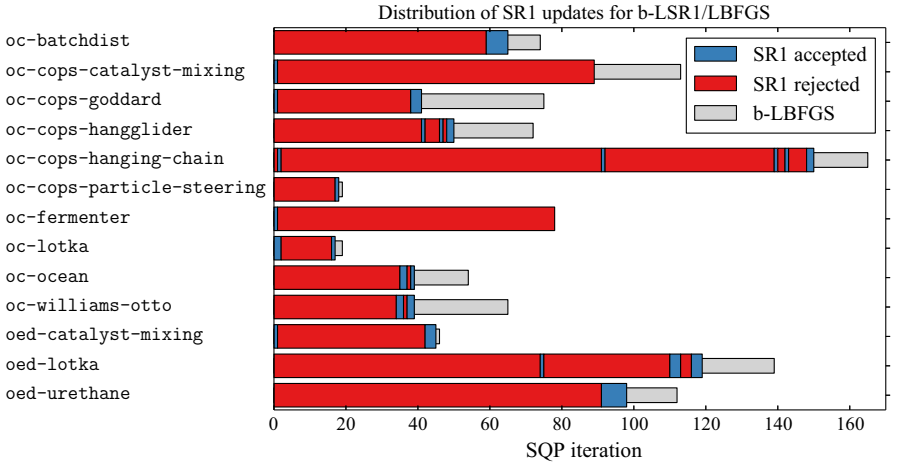

Fig. 3 Distribution of accepted and rejected b-LSR1 updates for the b-LSR1/LBFGS algorithm. The number of iterations for the BFGS algorithm are given for comparison

b-LBFGS. Note that this early termination does not impact the convergence of the SQP method, as the QP that delivers the SQP step is always solved to completion.

In practical situations, the dynamical systems are larger and more complicated than those considered here. The derivative evaluation, which is required for each SQP iteration, is then expected to dominate the computation time compared to the solution of the QPs. This is especially true for OED problems which require exact second-order

Table 5 Total number of QP iterations and overall computing times of the b-LSR1/LBFGS algorithm with and without early termination of the QP solution

Instance	With QP early term		Without QP early term	
	QP	Time (s)	QP	Time (s)
oc-batchdist	622	13.83	31,683	130.54
oc-cops-catalyst-mixing	1415	2.48	4410	4.31
oc-cops-goddard	313	0.75	7727	4.17
oc-cops-hangglider	417	1.73	56,407	33.91
oc-cops-hanging-chain	303	1.63	179,449	61.44
oc-cops-particle-steering	337	0.7	21,754	18.04
oc-fermenter	1367	6.5	20,873	48.64
oc-lotka	507	0.48	1525	1.05
oc-ocean	5167	2.06	23,171	8.69
oc-williams-otto	1750	3.75	7889	14.87
oed-catalyst-mixing	650	1.33	7828	5.04
oed-lotka	1738	4.92	16,591	21.07
oed-urethane	5669	549.33	260,394	1706.15

The number of SQP iterations is not affected

derivatives of the dynamics for the computation of the first derivative of (15d). We thus expect that our b-LSR1/LBFGS method will be superior in terms of CPU time for large dynamic systems.

7 Summary

We presented a filter line-search SQP algorithm that exploits block-diagonal structure of the Hessian matrix. This structure often occurs in the context of direct methods for optimal control. A blockwise approximation scheme based on the SR1 update and a scaled BFGS update was devised that uses the SR1 update whenever possible. The BFGS update is employed as fallback strategy. The resulting sparse and nonconvex quadratic subproblems are handled by a new Schur-complement variant of the parametric active-set code qpOASES. An inertia control technique that was added to the QP solver permits the efficient verification of the positive definiteness condition required by the globalization strategy. A small numerical case study and numerical experiments with 13 large benchmark problems from optimal control and optimum experimental design demonstrate the performance improvement gained by the new method in terms of the number of SQP iterations and CPU time.

Acknowledgments The authors thank two anonymous referees and an anonymous technical editor for their constructive and helpful comments.
D. Janka and C. Kirches were supported by DFG Graduate School 220 (Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences) funded by the German Excellence Initiative. D. Janka was supported by BASF SE within the junior research group experimental design. C. Kirches and S. Sager were supported by the German Federal Ministry of Education and Research program "Mathematics for Innovations in Industry and Service 2013–2016", Grant No. 05M2013-GOSSIP. Financial support by

the European Union within the 7th Framework Programme under Grant Agreement No 611909 is gratefully acknowledged. A. Wächter was supported in part by the National Science Foundation Grant DMS-1216920. Parts of this work were carried out during an appointment of D. Janka to Northwestern University.

References

1. Amestoy, P., Dollar, H.S., Reid, J.K., Scott, J.A.: An approximate minimum degree algorithm for matrices with dense rows. Technical Report RAL-TR-2007-020, Rutherford Appleton Laboratory (2007)
2. Bauer, I., Bock, H.G., Körkel, S., Schlöder, J.P.: Numerical methods for initial value problems and derivative generation for DAE models with application to optimum experimental design of chemical processes. Sci. Comput. Chem. Eng. **II**(2), 282–289 (1999)
3. Bauer, I., Bock, H.G., Schlöder, J.P.: DAESOL—a BDF-code for the numerical solution of differential algebraic equations. Preprint, IWR der Universität Heidelberg, SFB **359** (1999)
4. Bischof, C., Khademi, P., Mauer, A., Carle, A.: ADIFOR 2.0: automatic differentiation of fortran 77 programs. IEEE Comput. Sci. Eng. **3**(3), 18–32 (1996)
5. Bisschop, J., Meeraus, A.: Matrix augmentation and partitioning in the updating of the basis inverse. Math. Program. **13**(1), 241–254 (1977)
6. Bock, H.G.: Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen, volume 183 of Bonner Mathematische Schriften. Universität Bonn, Bonn (1987)
7. Bock, H.G., Plitt, K.J.: A multiple shooting algorithm for direct solution of optimal control problems. In: Proceedings of the 9th IFAC World Congress, pages 242–247, Budapest, 1984. Pergamon Press. Available at http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf
8. Bryson, A.E.: Applied Optimal Control: Optimization, Estimation and Control. CRC Press, Boca Raton (1975)
9. Byrd, R.H., Khalfan, H.F., Schnabel, R.B.: Analysis of a symmetric rank-one trust region method. SIAM J. Optim. **6**(4), 1025–1039 (1996)
10. Conn, A.R., Gould, N.I.M., Toint, PhL: Testing a class of methods for solving minimization problems with simple bounds on the variables. Math. Comput. **50**(182), 399–430 (1988)
11. Conn, A.R., Gould, N.I.M., Toint, PhL: Convergence of quasi-Newton matrices generated by the symmetric rank one update. Math. Program. **50**(1–3), 177–195 (1991)
12. Conn, A.R., Gould, N.I.M., Toint, PhL: LANCELOT: A Fortran Package for Large-scale Nonlinear Optimization (Release A). Volume 17 of Springer Series in Computational Mathematics. Springer, Heidelberg (1992)
13. Contreras, M., Tapia, R.A.: Sizing the BFGS and DFP updates: numerical study. J. Optim. Theory Appl. **78**(1), 93–108 (1993)
14. Cuthrell, J.E., Biegler, L.T.: Simultaneous optimization and solution methods for batch reactor control profiles. Comput. Chem. Eng. **13**(1), 49–62 (1989)
15. Diehl, M., Bock, H.G., Kostina, E.: An approximation technique for robust nonlinear optimization. Math. Program. **107**, 213–230 (2006)
16. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
17. Dolan, E.D., Moré, J.J., Munson, T.S.: Benchmarking optimization software with COPS 3.0. Argonne National Laboratory Technical Report ANL/MCS-TM-273 (2004)
18. Duff, I.S.: MA57—a code for the solution of sparse symmetric definite and indefinite systems. ACM Trans. Math. Softw. **30**(2), 118–144 (2004)
19. Ferreau, H.J., Bock, H.G., Diehl, M.: An online active set strategy to overcome the limitations of explicit MPC. Int. J. Robust Nonlinear Control **18**(8), 816–830 (2008)
20. Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., Diehl, M.: qpOASES 3.0: a parametric active-set algorithm for quadratic programming. Math. Program. Comput. **6**(4), 327–363 (2014)
21. Fletcher, R.: Stable reduced hessian updates for indefinite quadratic programming. Math. Program. **87**(2), 251–264 (2000)
22. Forbes, J.: Model structure and adjustable parameter selection for operations optimizations. PhD thesis, McMaster University, Hamilton, Canada (1994)
23. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: A Schur-complement method for sparse quadratic programming. Technical report, Stanford Univ., CA (USA). Systems Optimization Lab. (1987)

24. Gill, P.E., Wong, E.: Methods for convex and general quadratic programming. Technical report, University of California at San Diego (2013)
25. Gould, N.I.M., Toint, PhL: An iterative working-set method for large-scale nonconvex quadratic programming. Appl. Numer. Math. **43**(1), 109–128 (2002)
26. Griewank, A., Toint, PhL: Local convergence analysis for partitioned quasi-Newton updates. Numerische Mathematik **39**(3), 429–448 (1982)
27. Griewank, A., Toint, PhL: Partitioned variable metric updates for large structured optimization problems. Numerische Mathematik **39**(1), 119–137 (1982)
28. Janka, D.: Optimum experimental design and multiple shooting. Diplomarbeit, Universität Heidelberg, Heidelberg (2010)
29. Janka, D.: Sequential quadratic programming with indefinite Hessian approximations for nonlinear optimum experimental design for parameter estimation in differential-algebraic equations. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2015. Available at http://archiv.ub.uni-heidelberg.de/volltextserver/19170/
30. Khalfan, H.F., Byrd, R.H., Schnabel, R.B.: A theoretical and experimental study of the symmetric rank-one update. SIAM J. Optim. **3**(1), 1–24 (1993)
31. Körkel, S.: Numerische Methoden für Optimale Versuchsplanungsprobleme bei nichtlinearen DAE-Modellen. PhD thesis, Universität Heidelberg (2002)
32. Körkel, S., Potschka, A., Bock, H.G., Sager, S.: A multiple shooting formulation for optimum experimental design. (2008). (submitted to Mathematical Programming, under review)
33. Leineweber, D.B., Bauer, I., Schäfer, A.A.S., Bock, H.G., Schlöder, J.P.: An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (parts I and II). Comput. Chem. Eng. **27**, 157–174 (2003)
34. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, Berlin (2006). ISBN 0-387-30303-0 (hardcover)
35. Oren, S.S., Luenberger, D.G.: Self-scaling variable metric (SSVM) algorithms part i: criteria and sufficient conditions for scaling a class of algorithms. Manag. Sci. **20**(5), 845–862 (1974)
36. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is NP-hard. J. Glob. Optim. **1**(1), 15–22 (1991)
37. Plitt, K.J.: Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diplomarbeit. Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn (1981)
38. Powell, M.J.D.: The convergence of variable metric methods for non-linearly constrained optimization calculations. Nonlinear Program. **3**, 27–63 (1978)
39. Pukelsheim, F.: Optimal Design of Experiments. Classics in Applied Mathematics 50. SIAM (2006). ISBN 978-0-898716-04-7
40. Sager, S.: On the Integration of Optimization Approaches for Mixed-Integer Nonlinear Optimal Control. Universität Heidelberg, August (2011). Habilitationsschrift
41. Sager, S.: Sampling decisions in optimum experimental design in the light of Pontryagin's maximum principle. SIAM J. Control Optim. **51**(4), 3181–3207 (2013)
42. Sager, S., Bock, H.G., Diehl, M., Reinelt, G., Schlöder, J.P.: Numerical methods for optimal control with binary control functions applied to a Lotka–Volterra type fishing problem. In: Seeger, A. (ed.) Recent Advances in Optimization, Volume 563 of Lectures Notes in Economics and Mathematical Systems, pp. 269–289. Springer, Heidelberg (2009). ISBN 978-3-5402-8257-0
43. Vanderbei, R.J., Shanno, D.F.: An interior-point algorithm for nonconvex nonlinear programming. Comput. Optim. Appl. **13**, 231–252 (1999)
44. Vassiliadis, V.S., Sargent, R.W.H., Pantelides, C.C.: Solution of a class of multistage dynamic optimization problems. Parts 1. and 2. Indus. Eng. Chem. Res. **10**(33), 2111–2133 (1994)
45. Wächter, A., Biegler, L.T.: Line search filter methods for nonlinear programming: motivation and global convergence. SIAM J. Optim. **16**(1), 1–31 (2005)
46. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)