**Mathematical Programming Computation manuscript No.**
(will be inserted by the editor)

# blockSQP 2: exploiting structure to improve performance

Reinhold Wittmann · Sebastian Sager

Received: date / Accepted: date

**Abstract** One approach to solving optimal control problems is Bock's direct multiple shooting method. This method yields lifted nonlinear optimization problems (NLPs) with a specific block structure. Exploiting this structure via tailored optimization algorithms can be computationally beneficial. We propose such methods, primarily within the framework of filter line search sequential quadratic programming (SQP) methods and the solver `blockSQP`. A key aspect introduced by multiple shooting is the distinction between free and dependent variables. The dependent variables can be formally eliminated from the quadratic subproblems (QPs) through a process known as *condensing*. We provide the theoretical background for efficiently handling bounds on dependent variables by distinguishing between explicit and implicit constraints. The latter arise from the interplay of bounds implicitly implied by nonlinear dynamics and the continuity conditions of the lifted problem. Subsequently, we analyze the iteration behavior of `blockSQP` using various optimal control examples and present a tailored convexification strategy for indefinite QPs. Applying this strategy to condensed QPs yields further benefits in certain instances. Since scaling significantly influences numerical optimization, we study its effect and discuss its connection to sizing strategies and termination criteria. To make the optimization less sensitive to scaling, we introduce a termination scheme based on achieved progress and a scaling heuristic that balances derivatives with respect to free and dependent variables. Numerical tests demonstrate that our proposed methods offer advantages compared to both the initial version of `blockSQP` and the general-purpose solver `ipopt` on a number of benchmark problems.

**Keywords** Quasi-Newton · Sequential quadratic programming · Direct methods for optimal control · Structured quadratic programming · Condensing algorithm

**Mathematics Subject Classification (2020)** 49M37 · 90C30 · 90C53 · 90C55

✉ Reinhold Wittmann
E-mail: reinhold.wittmann@ovgu.de wittmann@mpi-magdeburg.mpg.de

Sebastian Sager
E-mail: sager@ovgu.de sager@mpi-magdeburg.mpg.de

Department of Mathematics, Otto von Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany
Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, Sandtorstraße 1, 39106 Magdeburg, Germany

## 1 Introduction

Sequential Quadratic Programming (SQP) is one of the most effective classes of algorithms for solving non-linear optimization problems with constraints (Nonlinear Programming, NLP). The method's origins lie in the early 1970s when concepts were developed to solve NLPs by iteratively solving a sequence of simpler Quadratic Subproblems (Quadratic Programming, QP). The theoretical breakthrough is based on the observation that applying the Newton method to the Karush-Kuhn-Tucker (KKT) conditions of the original NLP yields a step direction that precisely corresponds to the solution of the derived QP. Important early work, which laid the foundation for the modern SQP form, came from Fletcher [13] and Powell [29], who popularized the use of quasi-Newton approximations (specifically the BFGS formula) for the Hessian of the Lagrangian function. The work of Gill and Murray [15] later led to the development of robust and practical implementations for large-scale problems, particularly through techniques for handling inequality constraints and active sets. A core conceptual underpinning was provided by Robinson [32]. Leineweber, Bock and coworkers later developed SQP methods specifically tailored to Bock's direct multiple shooting approach for optimal control problems [25].

The `blockSQP` algorithm [23] was developed and implemented by Janka, Kirches, Sager, and Wächter. It is a filter line search SQP method as presented in [42, 44], utilizing blockwise quasi-Newton updates. It is built around the QP solver `qpOASES` [12], which can manage nonconvex QPs and obtain a solution provided that the quadratic form is positive definite on the null-space of certain active constraints. `blockSQP` is intended for solving dynamic optimal control and optimum experimental design problems discretized via Bock's direct multiple shooting method [6], which naturally results in a block-diagonal Hessian.

Here, we present three classes of recent extensions to the solver, resulting in version 2 and improved speed and consistency. First, we distinguish between free variables (e.g., discretized controls in multiple shooting) and those that depend on them through model conditions (e.g., state variables linked by continuity conditions). In the QPs of an SQP method, the dependent variables become linearly dependent on the free variables and can be formally eliminated through a condensing algorithm [2,6]. This yields a QP in fewer variables, specifically with the size of a single shooting parametrization [39]. A practical challenge with condensing is that bounds on dependent variables are transformed into linear constraints in the condensed QP. If numerous, these constraints may negatively affect performance. Here, we investigate when and how these constraints can be omitted from the QP to reduce its size. The main idea will be distinguishing between bounds originating from the OCP and those enforced by the underlying model. While the latter can be omitted in a single shooting approach, they must generally be enforced in multiple shooting. We show that these model-based bounds can be omitted from the QPs with only minor compromises to the theoretical convergence guarantees.

Second, we analyze the iteration behavior of `blockSQP` on various examples to observe how it depends on the structure [40] of the solution. Specifically, the iterates often show strong oscillations on singular arcs and slow switching point determination on problems with bang-bang arcs. Fast convergence only occurs once possibly indefinite Hessians or Hessian substitutes are accepted. This acceptance occurs after a varying number of iterations, which is highly sensitive to small algorithmic and problem parameter differences. Therefore, convexification strategies were previously implemented [21,24], though they did not provide a consistent benefit compared to the default SR1-BFGS scheme. We propose a tailored convexification strategy based on adding scaled identities to the Hessian to reduce the number of required SQP iterations and improve consistency, which we demonstrate in

numerical tests. Further improvements can be made for some instances if this strategy is applied to the condensed Hessian or, equivalently, free components only. A drawback is that additional QPs must be solved in each iteration, which can be mitigated by solving the QPs in parallel.

Third, the scaling of variables and functions significantly affects the performance of numerical optimization algorithms [43, Sect. 3.8]. We establish a connection between scaling, quasi-Newton updates, sizing strategies [9, 28], and termination criteria. For termination, we present a progress-based criterion that is not directly affected by the problem's scale. We propose a method to embed it in existing criteria, the filter line search, and heuristics. To obtain a favorable rescaling, we propose to balance the magnitude of derivatives with respect to free and dependent variables. This tends to improve the step direction for the free variables and subsequently the quasi-Newton BFGS updates. Numerical examples show improvements for several instances with low to no negative impact on most other instances. Furthermore, we analyze an instance where our scaling performs poorly and discuss approaches for deriving heuristics that work for a wider range of problems.

## 1.1 Contributions and Outline

The remainder of this article is laid out as follows. Section 2 establishes our framework of optimal control, direct multiple shooting, and quasi-Newton SQP. Section 3 follows with an introduction to condensing algorithms. We observe how they transform dependent variable bounds of a QP into constraints of the condensed QP. By distinguishing between bounds explicitly defined in the OCP and those arising implicitly from the underlying model, we show that the latter may be omitted from condensed QPs and propose alternative methods to manage these bounds. In Section 4 we shift our focus to the effects of possibly indefinite Hessian approximations in the context of multiple shooting optimal control. We propose convexification strategies to better exploit their benefits and demonstrate advantages and drawbacks on various optimal control examples. Finally, in Section 5 we consider the scale of problems and its connection to quasi-Newton updates, sizing strategies, and termination criteria. We present a termination scheme that is less sensitive to scaling and a heuristic for rescaling in a way that is favorable for the optimization. The numerical examples from the previous section demonstrate benefits and remaining shortcomings. Section 6 concludes the study with a comparison of our updated version of `blockSQP` to other NLP solvers and a discussion on managing remaining shortcomings of our methods. Finally, we provide an outlook on methods that could operate on iterates and the problem formulation outside the optimization algorithm.

## 2 The framework of optimal control, direct multiple shooting and quasi-Newton SQP

### 2.1 Optimal control problems and shooting methods

Given a time interval $[t_0, t_F]$, we consider optimal control problems (OCPs) in the form

$$\begin{aligned}
\min_{x(t),x_0,u(t),p} \quad & \int_{t_0}^{t_F} \mathscr{L}(x(t),u(t),p)dt + \mathscr{E}(x(t_F),p) \\
s.t. \quad & \dot{x}(t) = f(x(t),u(t),p) && \text{(ODE)} \\
& x(t_0) = x_0 && \text{(initial values)} \\
& lb_c \leq c(x(t),u(t),p) \leq ub_c && \text{(constraints)} \\
& lb_x \leq x(t) \leq ub_x && \text{(state bounds)} \\
& lb_u \leq u(t) \leq ub_u && \text{(control bounds)} \\
& lb_p \leq p \leq ub_p && \text{(parameter bounds)}
\end{aligned}$$

with sufficiently smooth functions. There are many possible extensions.

Direct methods search for a solution by first discretizing the OCP to a finite dimensional nonlinear program (NLP), and then applying nonlinear optimization. The discretization is performed as follows.

With a time grid $t_0 < t_1 < ... < t_N = t_F$, the controls $u(t)$ can be parameterized on the time points by variables $u_0, u_1, ..., u_N$ and approximated, e.g. piecewise constant. Let $u := [x_0, p, u_0, u_1, ..., u_N]$ include $x_0$ and $p$ such that $u$ collects all variables barring the differential states $x(t)$. The states are often managed using either single - or multiple shooting.

**Single shooting** [39]. The ODE-solution $x(t)$ is considered as a function $S(t, x_0, p, u_0, ..., u_N, p)$ and the bounds and constraints are evaluated only on the time grid. With the complete solution function

$$S(u) := [S(t_k, x_0, p, u_0, ..., u_N)]_{k=1,...,N} = [S(u)_k]_{k=1,...,N}, \tag{1}$$

the states $x$ are substituted for $S(u)$ in the OCP. This yields an NLP in the parameters, the discretized controls and the initial value.

**Multiple shooting** [6]. The states $x_k := x(t_k)$ $k = 1, ..., N$ on the time grid are added as optimization variables. The ODE is solved only over the intervals $[t_k, t_{k+1}]$ with solution $\bar{S}(t, t_k, x_k, u_k, p)$. The solution function is defined at $t_k$, $k = 1, ..., N$ for the respective start time $t_{k-1}$ as

$$S_k(x_k, u_k, p) := \bar{S}(t_{k+1}, t_k, x_k, u_k, p). \tag{2}$$

Continuity of these local solutions over $[t_0, t_F]$ is ensured via continuity conditions - the linear coupling

$$x_{k+1} = S_k(x_k, u_k, p),\ k = 0, ..., N-1. \tag{3}$$

To obtain a structured NLP, all parameters $p$ are localized to time points $t_k$ as variables $p_k$ with the condition $p_k = p_{k-1},\ k = 1, ..., N$. We consider them as part of the controls $u_k$, set $x := [x_1, ..., x_N]$, $u := [x_0, u_0, u_1, ..., u_N]$, $F_k(x, u) := x_k - S_{k-1}(x_{k-1}, u_{k-1})$ $k = 1, ..., N$ and write the continuity conditions as

$$F(x, u) := [F_k(x, u)]_{k=1,...,N} = 0. \tag{4}$$

The objective is approximated with quadratures to yield

$$\bar{h}(x,u) \coloneqq \sum_{k=0}^{N-1} q_k(x_k,u_k) + m(x_N,u_N).$$

This results in an NLP

$$\begin{aligned}
&\min_{x,u} \bar{h}(x,u) \\
&s.t.\ x_{k+1} = S_k(x_k,u_k) && k = 0,\dots,N-1 \\
&\quad lb_c \le c(x_k,u_k) \le ub_c && k = 0,\dots,N \\
&\quad lb_x \le x_k \le ub_x && k = 0,\dots,N \\
&\quad lb_u \le u_k \le ub_u && k = 0,\dots,N
\end{aligned} \tag{5}$$

in the parameters, discretized controls, the initial value and state variables. The state variables $x$ depend on $u$ through (4), which is why we refer to $x$ as *dependent* - and to $u$ as *free* variables.
We now consider the structure of this NLP in the context of SQP.

## 2.2 SQP methods

Sequential quadratic programming (SQP) methods solve NLPs, such as (5). They generate steps $d_x, d_u$ by solving quadratic subproblems (QPs) of the form

$$\min_{x,u} \frac{1}{2}[x,u]^T H[x,u] + [x,u]^T \nabla_{x,u}\bar{h}(\bar{x},\bar{u}) \tag{6}$$

$$s.t.\ x_{k+1} = \frac{\partial S_k(\bar{x}_k,\bar{u}_k)}{\partial x_k}x_k + \frac{\partial S_k(\bar{x}_k,\bar{u}_k)}{\partial u_k}u_k - (\bar{x}_{k+1} - S_k(\bar{x}_k,\bar{u}_k)) \tag{7}$$

$$lb_c - c(\bar{x},\bar{u}) \le \frac{\partial c(\bar{x},\bar{u})}{\partial x,u}\cdot[x,u] \le ub_c - c(\bar{x},\bar{u}) \tag{8}$$

$$lb_x - \bar{x} \le x \le ub_x - \bar{x} \tag{9}$$

$$lb_u - \bar{u} \le u \le ub_u - \bar{u} \tag{10}$$

which yield steps $d_x$, $d_u$ as their optimal solution. These steps are used in a filter line search [42], where the step size $\alpha$ is selected such that $\alpha\cdot d_x, \alpha\cdot d_u$ satisfies an acceptance criterion. Thus, a sequence of iterates $\bar{x}^{(j)}$, $\bar{u}^{(j)}$ towards a local solution is generated. If the QP is infeasible or the line search does not find a step, feasibility restoration is invoked to find a new iterate that is acceptable by having lower constraint violation. For a QP solution to exist and have descent properties, the quadratic form $H$ is required to be uniformly positive definite on the null space of the set of constraints that are active at both $\bar{x}^{(j)}$ and $\bar{x}^{(j)} + d_x$ [23, Sect. 2].
In the context of QPs, $\bar{x}$, $\bar{u}$ denote the NLP variables set to some value or iterate, $[x,u]/[\bar{x},\bar{u}]$ is the vector of combined state and control QP/NLP variables. The Lagrangian $L$ of the NLP (5) is defined as

$$\begin{aligned}
L(\bar{x},\bar{u},\lambda,\mu,\nu,\sigma) = \bar{h}(\bar{x},\bar{u}) &+ \sum_{k=0}^{N}(\sigma_k^T c(\bar{x}_k,\bar{u}_k) + \lambda_k^T\bar{x}_k + \mu_k^T\bar{u}_k) \\
&+ \sum_{k=0}^{N-1}\nu_{k+1}^T(\bar{x}_{k+1} - S_k(\bar{x}_k,\bar{u}_k)).
\end{aligned} \tag{11}$$

Here $\lambda, \mu, \nu, \sigma$ denote the Lagrange multipliers, which are computed from the QPs as well. The Hessian of the Lagrangian is $H \coloneqq \nabla^2_{x,u} L(\bar{x}, \bar{u}, \lambda, \mu, \nu, \sigma)$.

The QP/NLP variables $x, u \,/\, \bar{x}, \bar{u}$ are sorted according to the time intervals

$$[x,u] = x_0, u_0 \mid x_1, u_1 \mid \ \dots \ \mid x_N, u_N,$$

which causes the Hessian of the Lagrangian to become block-diagonal

$$\frac{\partial}{\partial \bar{x}_k, \bar{u}_k} \frac{\partial}{\partial \bar{x}_j, \bar{u}_j} L = 0 \text{ if } k \neq j$$

$$H = \nabla^2_{x,u} L = \begin{bmatrix} H_0 & & & \\ & H_1 & & \\ & & \ddots & \\ & & & H_N \end{bmatrix}. \tag{12}$$

As this Hessian is computationally expensive to calculate, approximations or substitutes are often used instead. We simply refer to these as Hessians and call $\nabla^2_{\bar{x},\bar{u}} L$ the *exact Hessian*. Accordingly, we consider quasi-Newton methods.

### 2.3 Quasi-Newton methods

Quasi-Newton methods compute a Hessian $H^{(j)}$ in each iteration $j$ by updating the Hessian $H^{(j-1)}$ of the previous iteration using secant information. With the current iterate $\bar{x}^{(j)}, \bar{u}^{(j)}$, the previous iterate $\bar{x}^{(j-1)}, \bar{u}^{(j-1)}$ and the current multipliers $\lambda, \mu, \nu, \sigma$, set

$$\begin{aligned} d &\leftarrow [\bar{x}^{(j)}, \bar{u}^{(j)}] - [\bar{x}^{(j-1)}, \bar{u}^{(j-1)}], \\ \gamma &\leftarrow \nabla_{\bar{x},\bar{u}} L(x^{(j)}, u^{(j)}, \lambda, \mu, \nu, \sigma) - \nabla_{\bar{x},\bar{u}} L(\bar{x}^{(j-1)}, \bar{u}^{(j-1)}, \lambda, \mu, \nu, \sigma). \end{aligned}$$

The update of $H^{(j-1)}$ is computed via an update formula, e.g. the symmetric rank 1 (SR1) update

$$H^{(j)} \leftarrow H^{(j-1)} + \frac{(\gamma - H^{(j-1)} d)(\gamma - H^{(j-1)} d)^T}{(\gamma - H^{(j-1)} d)^T d},$$

which may yield indefinite Hessians, or the rank 2 BFGS update

$$H^{(j)} \leftarrow H^{(j-1)} - \frac{H^{(j-1)} d d^T H^{(j-1)}}{d^T H^{(j-1)} d} + \frac{\gamma \gamma^T}{\gamma^T d}.$$

Both updates preserve positive definiteness if $\gamma^T d > 0$. This condition is enforced for BFGS via Powell's damping strategy [30] to ensure positive definiteness of the resulting Hessians. It defines

$$\begin{aligned} \theta &\leftarrow \begin{cases} \frac{(1-0.2) \cdot d^T H^{(j-1)} d}{d^T H^{(j-1)} d - d^T y} & d^T \gamma < 0.2 \cdot d^T H^{(j-1)} d \\ 1 & \text{else} \end{cases} \\ \bar{\gamma} &\leftarrow \theta \cdot \gamma + (1-\theta) \cdot H^{(j-1)} d \end{aligned} \tag{13}$$

and computes the BFGS update with $\bar{\gamma}$. In this work, BFGS always refers to this damped update, yielding positive definite Hessians. For $H^{(0)}$, a scaled identity is typically chosen

both for SR1 and BFGS. `blockSQP` utilizes the block-diagonal structure (12) by applying the quasi-Newton updates separately to each block with the corresponding section of $d$, $\gamma$. This yields an update of rank $N+1$ for SR1 or rank $2\cdot(N+1)$ for BFGS while preserving the block-diagonal structure.

Indefinite Hessians may not fulfill the definiteness requirements of the QP, and may thus not produce a step. Positive definite Hessians always fulfill the condition, but may be a poor approximation of the exact Hessian if it is indefinite, causing poor steps and - in case of BFGS - losing the benefit of block-wise updates. Therefore, `blockSQP` maintains two Hessians by default, a potentially indefinite first Hessian and a positive definite fallback Hessian. QP solution is attempted with the first Hessian. If the definiteness condition is unfulfilled, the step is computed with the fallback Hessian instead. We will review this topic in Section 4.

## 3 Structured QPs, condensing and bound handling

In this section we provide an introduction to QP condensing and investigate methods to manage state bounds of the condensed QP. We achieve this by distinguishing between bounds demanded in the OCP and bounds originating from an underlying model.

As multiple shooting introduces various additional variables, solving the QPs may be time consuming. To reduce this cost, condensing [5] can be applied to the QPs to eliminate the additional variables and obtain smaller QPs that are cheaper to solve. The solution of the original QP is subsequently recovered from the solution of the condensed QP. The aim is to reduce the cost to that of a QP in single shooting.

### 3.1 Condensing procedure and implementation

In the QP, the dependent variables $x$ become linearly dependent on the free variables $u$ through the linearized continuity conditions (7). The condensing procedure begins by solving (7) for the states $x$. Writing these equations in terms of $F$ (4), we obtain

$$\frac{\partial F(\bar{x},\bar{u})}{\partial x}\cdot x = -\frac{\partial F(\bar{x},\bar{u})}{\partial u}\cdot u - F(\bar{x},\bar{u}).$$

This equation can be rearranged to

$$x = -\frac{\partial F}{\partial x}^{-1}\frac{\partial F}{\partial u}\cdot u - \frac{\partial F}{\partial x}^{-1}F(.) =: G\cdot u + g, \tag{14}$$

allowing us to eliminate the dependent variables $x$ from the QP. We separate the linear term

$$[x,u]^T\nabla_{x,u}\bar{h}(\bar{x},\bar{u}) =: u^T r + x^T q \tag{15}$$

and the quadratic term

$$\frac{1}{2}[x,u]^T H[x,u] =: \frac{1}{2}(x^T Rx + u^T Qu + x^T Su + u^T S^T x). \tag{16}$$

By inserting (14) into (15) and (16), we obtain a QP in $u$ with the linear form

$$h := r + G^T(Qg+q) + S^T g$$

and the condensed Hessian

$$H \coloneqq R + G^T Q G + G^T S + S^T G \tag{17}$$

as the quadratic form. The algorithms employed for the above computations are described in [2] and have been added to `blockSQP`. Our implementation can handle several differential systems coupled by constraints and discretized in turn via multiple shooting. A shortcoming is that we cannot output condensed QPs in dense format because currently, `qpOASES` only supports nonconvex QPs if they are in sparse format. As a consequence, condensing currently benefits only problems with a large number of states, hence we will not activate it for the problems from Appendix B. For this Section, we instead consider the P2Chem problem outlined in Appendix D. It is a model with 5 systems of differential-algebraic equations coupled linearly by constraints. Its full and condensed constraint Jacobian are shown in Fig. 1 below.

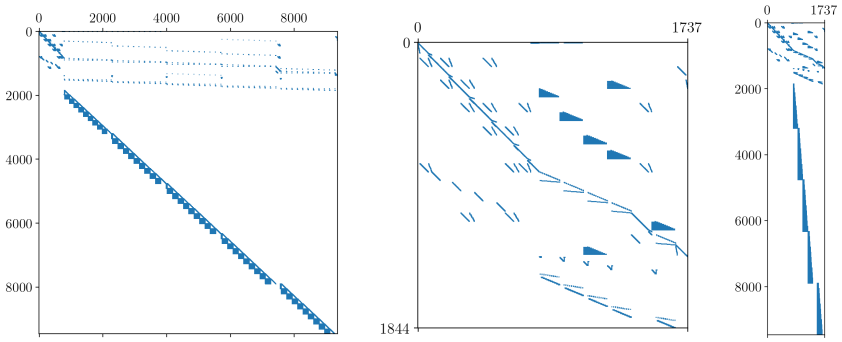



**Fig. 1** Structural nonzero elements of different matrices. Left: The five lower block-diagonal submatrices of the constraint Jacobian of the P2Chem problem correspond to the linearized continuity conditions of the five DAE-systems. Middle: the corresponding condensed constraint matrix. It is smaller in dimension, but less structured. Right: the same matrix with additional rows corresponding to state bounds.

Condensing reduces the number of variables, but the number of constraints stays the same if state bounds are included. This is because the continuity conditions are used in the elimination of the state variables, but the state bounds (9) turn into linear constraints

$$lb_x - \bar{x} \leq x \leq ub_x - \bar{x} \quad \rightarrow \quad lb_x - \bar{x} \leq -\frac{\partial F}{\partial x}^{-1} \frac{\partial F}{\partial u} \cdot u - \frac{\partial F}{\partial x}^{-1} \cdot F(.) \leq ub_x - \bar{x}. \tag{18}$$

These constraints can contribute to the QP becoming infeasible as well. Therefore, it is important to investigate alternative methods to manage these bounds. Our goal is to have condensed QPs be comparable in size to QPs in single shooting.

### 3.2 Shooting, condensing and implicit bounds

As we have observed in the previous Section, including state bounds as linear constraints can considerably increase the size of the condensed QP. If these bounds are demanded as part of the OCP, then adding them is necessary if one wishes to retain the convergence properties of

SQP. This is true for single shooting as well, where they have to be included in the NLP as constraints $lb_x \leq S(t_k,u) \leq ub_x$, compare Sect. 2.1. Thus, they turn into linear constraints in the QP which, as a result, is of a similar size as the condensed QP in multiple shooting. The situation is different; however, if the bounds originate from the underlying model equations. We restrict ourselves to ODE models.

**Definition 1 (Implicit bounds)**
Consider an ODE

$$\dot{x} = f(x(t),u(t)),\ t \in [t_0,t_F].$$

If for any feasible control $u(t)$ and any $t_s \in [t_0,t_F)$ with an initial value $x_s$ it holds that

$$lb_x \leq x_s \leq ub_x \ \Rightarrow\ \forall t \in (t_s,t_F]:\ lb_x \leq x(t) \leq ub_x,$$

then we refer to $lb_x$, $ub_x$ as *implicit bounds* for $x(t)$, $t \in [t_0,t_F]$.

Since controls may be restricted to be constant, Definition 1 encompasses ODEs with parameters as well.

*Example 1* Consider the ODE

$$\dot{x}(t) = (1-x(t))^2 \cdot p,\ x(0) = x_0,\ -\infty < x_0 \leq 1,\ p \geq 0$$

where any solution has the representation

$$x(t) = 1 - \frac{1}{\frac{1}{1-x_0} + p \cdot t} \ \text{ if } \ x_0 \neq 1$$

or $x(t) \equiv 1$ if $x_0 = 1$. The ODE yields $x(t) \leq 1$ for any initial value $-\infty < x_s < 1$ at any time point $t_s \geq 0$ and therefore enforces the implicit bounds

$$-\infty \leq x(t) \leq 1, \quad t \geq 0.$$

If $x_0 > 1$ and $p > 0$, then the solution only exists for $t < -\frac{1}{p(1-x_0)}$.

Example 1 shows that the ODE solution may not exist up to $t_F$ if an initial value violates the implicit bounds. In the following, we do not assume existence of any ODE solution up to any time point $t > t_s$ if an initial value $x_s$ at time $t_s$ violates implicit bounds. In the context of (4), we assume that $F$ cannot be evaluated if a stage state violates an implicit bound.

*Example 2* In the P2Chem problem in Appendix D, the differential states are mole fractions, which naturally have the implicit lower bound 0 and possibly - depending on modeling and implementation details - the implicit upper bound 1. Several problems from Appendix B such as Lotka Volterra fishing, generate implicit lower bounds of 0 as well.

In the context of OCPs, we say the states $x$ are subject to implicit bounds $lb_x \leq x(t) \leq ub_x$ if $lb_x$, $ub_x$ satisfy Definition 1 and are demanded for the initial value $x_0$.

If single shooting is used, the states are substituted for the solution function $S(\bar{u})$, which per Definition (1) always satisfies implicit bounds

$$lb_x \leq S(\bar{u})_k \leq ub_x,\ k = 1,...,N.$$

Consequently, we may omit them from the NLP.

If multiple shooting is used, the implicit bounds are enforced only when all continuity conditions $x_{k+1} = S_k(x_k,u_k)$ hold. As they may intentionally be violated during the optimization iterations, the stage states $x_{k,\ k=1,...,N}$ may violate implicit bounds. In that case, we do not assume that the ODE model is valid or can be evaluated at all; thus we must include these bounds in the NLP and enforce them for every iterate. In the next Section we investigate ways to achieve this while omitting them from condensed QPs.

### 3.3 Truncation and quantification of model bound violations

**Notation.** $x$, $u$ now denote NLP variables/iterates, $d_x$, $d_u$ denote QP variables or NLP steps.

If we exclude implicit bounds from the condensed QPs, we have to enforce them some other way. Assume that at some point $x,u$ it holds that $lb_x \leq x \leq ub_x$. Let $d = d_x$, $d_u$ be a step calculated by solving a QP without implicit bounds and let $lb_x \leq x + d_x \leq ub_x$ be violated. To accept $d$, a correction $v_x$ has to be added to $d_x$ such that

$$lb_x \leq x + d_x + v_x \leq ub_x.$$

Straightforward is truncation

$$v_x \leftarrow v_x^{(t)} := \max\{lb_x - x - d_x, 0\} + \min\{ub_x - x - d_x, 0\} \tag{19}$$

where min and max are considered component-wise. However, it is important to determine whether it affects the theoretical convergence properties of the SQP method. Because model bound violation is the result of continuity violation, another idea is to apply a second order correction (SOC) [8, 15.3.2.3] to the continuity conditions to reduce both their violation and the model bound violation. Normally, this would require evaluating $F$ in (4) at $x + d_x, u + d_u$ which we assumed is not possible, thus it remains unclear how one would define it. We examine both issues in the following.

**Basic assumptions.** We first consider the global convergence analysis performed in [42]. Several assumptions are made for the equality constrained case. The following are of interest to us and are made for iterations in which no feasibility restoration is invoked.

- (G1) The NLP iterates $\xi^{(j)}$ along with $\xi^{(j)} + \alpha \cdot \bar{d}$ for $\alpha \in [0,1]$ and $\bar{d}$ being the step from the QP are contained in an open set $\bar{C} \subset \mathbb{R}^n$. The objective and constraints as well as their first derivatives are assumed to be bounded and Lipschitz-continuous over $\bar{C}$.
- (G2) The matrices approximating the Hessian of the Lagrangian are uniformly bounded.

The inequality constrained case is treated similarly, with (G1) and (G2) carrying over.
For simplicity, we consider an equality constrained NLP in the generalized form

$$\begin{aligned} \min_{x,u} \quad & \bar{h}(x,u) \\ s.t. \quad & c(x,u) = 0 \\ & F(x,u) = 0 \qquad (20) \\ & lb_x \leq x \leq ub_x \qquad (21) \end{aligned}$$

where $F : \{x \in \mathbb{R}^{n_x} | \, lb_x \leq x \leq ub_x\} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and (20) implicitly defines $x$ as a function $S(u)$, $S : \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. (21) are assumed to be implicit bounds with $lb_x \leq S(u) \leq ub_x$ for all $u$. This corresponds to an NLP (5) with fixed initial values satisfying the implicit bounds. In case $F$ and $S$ are as in (4) and (1), we say $F$ has shooting structure.

In the following, we denote with $x,u$ the NLP variables or some iterate of them and require them to fulfill the bounds (21). As we can - in general - not evaluate $F(x + \alpha d_x, u + \alpha d_u)$, we modify assumption (G1) accordingly to

- (G1') The objective and constraints as well as their first derivatives are bounded and Lipschitz-continuous over an open set $\tilde{C}$, with the convex combinations $\left[[x^{(j)}, u^{(j)}], [x^{(j)} + d_x^{(j)}, u^{(j)} + d_u^{(j)}]\right]$ as well as $\left[[x^{(j)}, u^{(j)}], [S(u^{(j)}), u^{(j)}]\right]$ being contained in the set

$$C := \{[x,u] \in \tilde{C} : lb_x \leq x \leq ub_x\}.$$

Another consequence is that [42, Lemma 3] cannot be formulated.
Let $\bar{\theta}$ be some norm of the constraint violation and $\bar{h}$ the objective. With a normally calculated step $\bar{d}$ and under assumption (G1), it states that

$$|\bar{\theta}(\xi^{(j)} + \alpha\bar{d}) - (1-\alpha)\bar{\theta}(\xi^{(j)})| \leq C_\theta \alpha^2 \|\bar{d}\|^2 \tag{22}$$

$$|\bar{h}(\xi^{(j)} + \alpha\bar{d}) - \bar{h}(\xi^{(j)}) - \alpha\nabla\bar{h}(\xi^{(j)})^T\bar{d}| \leq C_{\bar{h}} \alpha^2 \|\bar{d}\|^2 \tag{23}$$

for suitable constants $C_\theta, C_{\bar{h}}$. We discuss the implications in Section 3.4.

To circumvent the need to evaluate $F(x+d_x, u+d_u)$, we require a different measure for the violation of the continuity conditions.

**Definition 2 (Residual formulation)**
We call

$$R(x,u) \coloneqq x - S(u)$$

the residual with the condition

$$R(x,u) = 0$$

as the residual formulation of (20).

To perform our analysis, we require the following statements.

a) The implicit function $S$ can be defined over all $u$.
b) $\frac{\partial F}{\partial x}$ is invertible for all $[x,u] \in C$ such that $S$ and $R$ are continuously differentiable by the implicit function theorem.
c) The singular values of $\frac{\partial F}{\partial x}$ are uniformly bounded below such that $\frac{\partial F}{\partial x}^{-1}$ is uniformly bounded.
d) There exist constants $M_F$, $M_R > 0$ such that for all $[x,u] \in C$

$$\|F(x,u)\| \leq M_F \cdot \|R(x,u)\|, \tag{24}$$

$$\|R(x,u)\| \leq M_R \cdot \|F(x,u)\|. \tag{25}$$

In the context of multiple shooting, a) states that solving the ODE should be possible for any control and feasible initial values, with b) following directly from the structure of $F$ and (G1). Statement c) mirrors assumption (G4) from [42]. The existence of $M_F$ for (24) follows from assumption (G1'), as $F$ being Lipschitz-continuous with constant $\bar{L}$ yields

$$\|F(x,u)\| = \|F(x,u) - F(S(u),u)\| \leq \bar{L}\|x - S(u)\| = \bar{L}\|R(x,u)\|.$$

Showing the existence of $M_R$ for (25) is more involved and requires the assumptions that all $[x,u]$ are in a compact set $\mathscr{C} \subset C$ and c). As an alternative, we can assume that $F$ has shooting structure. (G1') ensures the solution functions $S_k$ in (2) are Lipschitz-continuous as well with constant $\bar{L}$, and we have for $k \geq 2$

$$\begin{aligned}
\|R(x,u)_k\| &= \|x_k - S(u)_k\| = \|x_k - S_{k-1}(S(u)_{k-1}, u_{k-1})\| \\
&= \|x_k - S_{k-1}(x_{k-1}, u_{k-1}) + S_{k-1}(x_{k-1}, u_{k-1}) - S_{k-1}(S(u)_{k-1}, u_{k-1}))\| \\
&\leq \|F(x,u)_k\| + \bar{L} \cdot \|x_{k-1} - S(u)_{k-1}\| \\
&= \|F(x,u)_k\| + \bar{L} \cdot \|R(x,u)_{k-1}\| \\
&\Rightarrow \|R(x,u)_k\| \leq \sum_{j=1}^{k} \bar{L}^{k-j} \cdot \|F(x,u)_j\| \quad k = 1,\ldots,N \\
&\Rightarrow \|R(x,u)\| \leq M_R \cdot \|F(x,u)\|
\end{aligned}$$

for a suitable $M_R > 0$. Note that $F(x,u)_1 = R(x,u)_1 = x_1 - S(u)_1$.

We need to quantify the model bound violation $v$ at $x+d_x$ with respect to the step length $\|d\|$. To not interfere with the local convergence theory presented in [44], we require $v = o(\|d\|)$. We show some relations between $F$ and $R$ in preparation, considering the above assumptions. We implicitly utilize the Lipschitz conditions to quantify terms as $O(.)$.

**Lemma 1** *Let $\frac{\partial F}{\partial x}(x,u)^{-1}$ be uniformly bounded for $[x,u] \in C$. Then*

$$R(x,u) = \frac{\partial F}{\partial x}(x,u)^{-1} \cdot F(x,u) + O(\|R(x,u)\|^2). \tag{26}$$

*Proof*

$$\begin{aligned} 0 &= F(S(u),u) = F(x-(x-S(u)),u) = F(x-R(x,u),u) \\ &= F(x,u) - \frac{\partial F}{\partial x}(x,u) \cdot R(x,u) + O(\|R(x,u)\|^2) \\ &\Rightarrow R(x,u) = \frac{\partial F}{\partial x}(x,u)^{-1} \cdot F(x,u) + O(\|R(x,u)\|^2) \end{aligned}$$

□

**Lemma 2** *Let $\frac{\partial F}{\partial x}(x,u)^{-1}$ and $\frac{\partial F}{\partial u}(x,u)$ be uniformly bounded and Lipschitz-continuous for $[x,u] \in C$. Subsequently, it holds for the derivative of $R$ that*

$$\frac{\partial R}{\partial u}(x,u) = \frac{\partial F}{\partial x}(x,u)^{-1} \cdot \frac{\partial F}{\partial u}(x,u) + O(\|R(x,u)\|). \tag{27}$$

*Proof* From the implicit function theorem, we can deduce

$$\begin{aligned} \frac{\partial R}{\partial u}(x,u) &= -\frac{\partial}{\partial u} S(u) \\ &= \frac{\partial F}{\partial x}(S(u),u)^{-1} \cdot \frac{\partial F}{\partial u}(S(u),u) \\ &= \frac{\partial F}{\partial x}(x-R(x,u)),u)^{-1} \cdot \frac{\partial F}{\partial u}(x-R(x,u),u) \\ &= (\frac{\partial F}{\partial x}(x,u)^{-1} + O(\|R(x,u)\|) \cdot (\frac{\partial F}{\partial u}(x,u) + O(\|R(x,u)\|)) \\ &= \frac{\partial F}{\partial x}(x,u)^{-1} \cdot \frac{\partial F}{\partial u}(x,u) + O(\|R(x,u)\|). \end{aligned}$$

□

Note that the additional assumption $\frac{\partial F}{\partial x}(x,u)^{-1}$ Lipschitz follows from (G1') if $F$ has shooting structure or from (G1') and c) if all $[x,u]$ are assumed to be in a compact set.

We show that an SQP step $d$ reduces the residual $R$ down to second order respective to $d$. Hence, $R$ can be used as a substitute for measuring the continuity condition and model bound violation.

**Lemma 3** *Let $d = d_x, d_u$ be a step calculated by solving a QP, which satisfies in particular*

$$\frac{\partial F}{\partial x}(x,u)\cdot d_x + \frac{\partial F}{\partial u}(x,u)\cdot d_u + F(x,u) = 0. \tag{28}$$

*and let assumptions (G1') and (G2) hold. Let $\frac{\partial F}{\partial x}(x,u)^{-1}$ be uniformly bounded and Lipschitz continuous. Then*

$$R(x+d_x, u+d_u) = O(\|d\|^2). \tag{29}$$

*Proof* The implicit function theorem ensures that $R = x - S(u)$ is differentiable as often as $F$, and its derivatives are bounded and Lipschitz continuous. Using Lemma 1 and 2 we obtain

$$\begin{aligned}
&R(x+d_x,u+d_u) = R(x,u) + d_x + \frac{\partial R}{\partial u}(x,u)\cdot d_u + O(\|d\|^2)\\
&= R(x,u) + d_x + \frac{\partial F}{\partial x}(x,u)^{-1}\cdot F(x,u) - \frac{\partial F}{\partial x}(x,u)^{-1}\cdot F(x,u)\\
&\quad + \frac{\partial R}{\partial u}(x,u)\cdot d_u - \frac{\partial F}{\partial x}(x,u)^{-1}\cdot\frac{\partial F}{\partial u}(x,u)\cdot d_u + \frac{\partial F}{\partial x}(x,u)^{-1}\cdot\frac{\partial F}{\partial u}(x,u)\cdot d_u + O(\|d\|^2)\\
&\overset{(28)}{=} R(x,u) - \frac{\partial F}{\partial x}(x,u)^{-1}\cdot F(x,u) + \frac{\partial R}{\partial u}(x,u)\cdot d_u - \frac{\partial F}{\partial x}(x,u)^{-1}\cdot\frac{\partial F}{\partial u}(x,u)\cdot d_u + O(\|d\|^2)\\
&\overset{(26)}{=} O(\|R(x,u)\|^2) + (\frac{\partial R}{\partial u}(x,u) - \frac{\partial F}{\partial x}(x,u)^{-1}\cdot\frac{\partial F}{\partial u}(x,u))\cdot d_u + O(\|d\|^2)\\
&\overset{(27)}{=} O(\|R(x,u)\|^2) + O(\|(R(x,u)\|\cdot\|d\|) + O(\|d\|^2).
\end{aligned}$$

(25) shows that $O(\|R(x,u)\|) = O(\|F(x,u)\|)$. Assumptions (G1') and (G2) ensure the KKT-matrix [42, (3)] is uniformly bounded; thus $\|d\| \geq m_F\|F(x,u)\| \Leftrightarrow \|F(x,u)\| \leq \frac{1}{m_F}\|d\|$ for some constant $m_F > 0$. Therefore, $R(x,u) = O(\|d\|)$ and together with the above

$$R(x+d_x,u+d_u) = O(\|d\|^2).$$

□

### 3.4 Truncation and global convergence of SQP

Because $lb_x \leq S(.) \leq ub_x$, we have

$$x + d_x - lb_x \geq x + d_x - S(u+d_u) = R(x+d_x,\, u+d_u) \tag{30}$$
$$x + d_x - ub_x \leq x + d_x - S(u+d_u) = R(x+d_x,\, u+d_u). \tag{31}$$

$\Rightarrow$ Lower/upper model bound violations are bounded below/above by $R(x+d_x,u+d_u)$.

Equations (29), (30), and (31) show that the truncation (19) is of order $O(\|d\|^2)$. We can thus correct any model bound violation with a step $v$ of order $O(\|d\|^2)$. One can verify that the local convergence results in [44] hold if a lower order step is added to the full step, provided the original step is used to define the switching and Armijo conditions. Local superlinear convergence is shown by proving that full steps are accepted near local optima. Compare how second order correction steps are added into the filter line search.

A key difference is that the corrections must be applied to reduced steps $\alpha\cdot d$, $0 < \alpha \leq 1$ of each line search iteration, unlike SOC, which is only applied to the first. We denote

these steps with $v_x(\alpha)$, and for truncation it holds $v_x^{(t)}(\alpha) = O(\alpha\|d\|^2)$. Although we can reformulate assumption (G1) with the statement that $[x+\alpha\cdot d_x + v_x(\alpha), u+\alpha\cdot d_u] \in \bar{C}$, Lemma 3 of [42] will not hold as we will need to restrict the order of the corrections to $v_x(\alpha) = O(\alpha^2\|d\|^2)$. In [42], it is used in the following way to show global convergence. First the filter condition ensures that [42, Theorem 1]

$$\lim_{j\to\infty} \bar{\theta}(\xi^{(j)}) = 0.$$

Subsequently, for $\bar{\theta}$ sufficiently small, (22) and (23) are used in the proof of there existing a step size $\bar{\alpha}$ depending only on $\bar{\theta}(\xi^{(j)})$ and fixed parameters, below which the Armijo condition holds and the trial iterate is acceptable for the filter [42, Lemma 7-11]. This result along with the regularity assumptions (G1-5), is used to show convergence to a point satisfying the optimality conditions [42, Theorem 2]. Without (22, 23), the filter line search is no longer guaranteed to find a step size acceptable to the filter and satisfying the Armijo condition close to the feasible region, thus global convergence no longer holds. In the iterations, this would manifest itself as the filter line search failing arbitrarily close to the feasible region or the step size becoming arbitrarily small. It is possible to remedy this by falling back to a QP with added implicit bounds when this occurs. We choose to forego this and always revert to the feasibility restoration phase or some heuristic upon line search failure. Similar compromises had already been made in the original implementation by not employing the step size formula [42, (18)] in favor of a fixed minimum step size or by not guaranteeing uniformly positive definite reduced Hessians [23, Sect. 2]. In practice, allowing the step size to become too small risks stalling the iterations, and reverting to a feasibility restoration phase can often be preferable in such situations.

Therefore, in most practical settings with few to no line search failures or when doing no fallback, the cost of solving the condensed QPs becomes comparable to that of solving the QPs in single shooting. Note that owing to the redundancy of the implicit bounds, we do not require their associated Lagrange multipliers and can fix them to zero.

### 3.5 Alternative ways to correct model bound violations

Far away from a local optimum where steps are large, truncation may significantly change the states $x$, which may increase both the objective function and the constraint violation. Thus, we consider other strategies to enforce the implicit bounds, starting with second order correction. Rather than trying to perform SOC for $F$, (24) and (25) allow us to define the second order correction for $R$. This is further justified by the bounds (30),(31) on the model bound violation.

**Lemma 4** *Let $d_x^{(c)}, d_u^{(c)}$ satisfy*

$$\frac{\partial F(x,u)}{\partial x} R(x+d_x, u+d_u) + \frac{\partial F(x,u)}{\partial x} d_x^{(c)} + \frac{\partial F(x,u)}{\partial u} d_u^{(c)} = 0. \tag{32}$$

*Then $d_x^{(c)}$, $d_u^{(c)}$ are a second order correction with respect to the residual $R$.*

*Proof*

$$
\begin{aligned}
&R(x+d_x+d_x^{(c)},\, u+d_u+d_u^{(c)}) \\
&= R(x+d_x,\, u+d_u) + d_x^{(c)} + \frac{\partial R(x+d_x, u+d_u)}{\partial u}\cdot d_u^{(c)} + O(\|d_u^{(c)}\|^2) \\
&= R(x+d_x,\, u+d_u) + d_x^{(c)} + \frac{\partial R(x,u)}{\partial u}\cdot d_u^{(c)} + O(\|d\|\cdot\|d_u^{(c)}\|) + O(\|d_u^{(c)}\|^2) \\
&\overset{(27)}{=} R(x+d_x,\, u+d_u) + d_x^{(c)} + \frac{\partial F(x,u)}{\partial x}^{-1}\frac{\partial F(x,u)}{\partial u}\cdot d_u^{(c)} \\
&\qquad\qquad + O(\|R(x,u)\|\cdot\|d_u^{(c)}\|) + O(\|d\|\cdot\|d_u^{(c)}\|) + O(\|d_u^{(c)}\|^2) \\
&= R(x+d_x,\, u+d_u) + d_x^{(c)} + \frac{\partial F(x,u)}{\partial x}^{-1}\frac{\partial F(x,u)}{\partial u}\cdot d_u^{(c)} + O(\|d\|^3) \qquad (33) \\
&\overset{(32)}{=} O(\|d\|^3)
\end{aligned}
$$

□

The rest of the SOC-QP can be chosen as in [44, Section 2] or [8, 15.3.2.3]. $d^{(c)} = O(\|d\|^2)$ holds as in the classical second order correction. As in [44], several second order corrections may be applied consecutively and considered as one second order correction.

Although the second order correction is done based on both the objective and constraints, it targets all continuity condition violations. This is in contrast to the idea of multiple shooting, which explicitly allows the continuity to be violated. Thus, we apply only minimal corrections inside the QP, targeting only the model bound violations. With $\bar{x}, \bar{u}$ now being an NLP iterate and $x, u$ denoting the optimization variables of the QP, our method is to add the truncation step $v_x^{(t)}$ to $x$. With $x$ and consequently $d_x$ being expressed by (14), we obtain the modified formula

$$x = -\frac{\partial F(\bar{x},\bar{u})}{\partial x}^{-1}\frac{\partial F(\bar{x},\bar{u})}{\partial u}\cdot u - \frac{\partial F(\bar{x},\bar{u})}{\partial x}^{-1} F(\bar{x},\bar{u}) + v_x^{(t)}.$$

We use it in the condensing procedure to obtain a modified condensed QP. Solving yields a corrected step $d_x + d_x^{(c)}, d_u + d_u^{(c)}$, which satisfies

$$
\begin{aligned}
d_x + d_x^{(c)} &= -\frac{\partial F(\bar{x},\bar{u})}{\partial x}^{-1}\frac{\partial F(\bar{x},\bar{u})}{\partial u}(d_u + d_u^{(c)}) - \frac{\partial F(\bar{x},\bar{u})}{\partial x}^{-1} F(\bar{x},\bar{u}) + v_x^{(t)} \\
&\Leftrightarrow \\
d_x^{(c)} &= -\frac{\partial F(\bar{x},\bar{u})}{\partial x}^{-1}\frac{\partial F(\bar{x},\bar{u})}{\partial u} d_u^{(c)} + v_x^{(t)}.
\end{aligned}
$$

As above, $d_x^{(c)}, d_u^{(c)} = O(\|d\|^2)$. The correction is not guaranteed to eliminate the model bound violation. By substituting in (33), it yields

$$R(\bar{x}+d_x+d_x^{(c)}, \bar{u}+d_u+d_u^{(c)}) = R(\bar{x}+d_x, \bar{u}+d_u) + v_x^{(t)} + O(\|d\|^3).$$

For $\|d\|$ small, $v_x^{(t)}$ increases negative components of $R$ and decreases positive ones. As with second order correction, we can repeat this minimal correction up to a fixed number of times. Owing to the bounds (30), (31), we can expect without guarantee that the model bound violation is reduced for $\|d\|$ small. Continuing the minimalistic approach, we treat

this correction analogous to the second order correction in [44], only appyling it in the first iteration of the line search or else falling back to truncation. This is to avoid introducing correction in cases where the step size reduction already prevented implicit bounds from being violated.

### 3.6 Implementation in `blockSQP`

Because SQP in general enforces variable bounds in each iteration, we ensure this holds exactly by truncating any bound violation originating from numerical errors in the QP solution procedure. As such, model bound violations are truncated by default. We implemented the minimal bound correction strategy discussed above. In our numerical experiments, it usually eliminated model bound violation within our set maximum number of six repeated corrections, even far away from any local optimum. Because `qpOASES` is able to hotstart the QP solution procedure from the previous solution, these additional QPs incur minor additional cost. We experimented with adding only the implicit bounds that were violated to the QP and re-solving it. Because this changes the QP size and `qpOASES` cannot hotstart from a differently sized QP, this strategy performed rather poorly and has not been used further. It additionally carries the risk of making QPs infeasible. As we only have one example of a problem with many states with implicit bounds as of right now, we cannot suggest the ideal method of handling bounds. We believe it worthwhile to consider them for future problems.

## 4 Indefinite Hessians in optimal control

In this section we investigate the effects of indefinite Hessians in the context of optimal control, beginning with case studies of example problems with varying solution structure. "Indefinite" in this work refers to potentially indefinite. We propose a convexification scheme with the goal of making them get accepted by the algorithm more often.

As mentioned, `blockSQP` maintains two Hessians by default, an indefinite one via SR1 quasi-Newton updates and a positive definite fallback Hessian via damped BFGS updates. In [23, Sect. 1.1] the importance of allowing indefinite Hessians was demonstrated on the example

$$\min_{x,y} x^2 - 0.5y^2 \;\; s.t.\; x - y = 0, \quad x^{(0)}, y^{(0)} = 10, 10$$

where block-wise BFGS performed significantly worse than block-wise SR1. Similarly, we consider indefinite SR1 and BFGS Hessians in the context of optimal control.

If the exact Hessian is available cheaply, then it may be passed to `blockSQP` and used in place of SR1, and one can then expect local quadratic convergence [8, Ch. 15], [44]. However, the exact Hessian becomes expensive to calculate for large problems. Because a benefit of `blockSQP` are the block-wise updates, we only consider SR1.

### 4.1 Optimal control case studies

For most problems, only the fallback BFGS Hessian is accepted far away from a local optimum, while indefinite Hessians start being accepted close to a local optimum [23, Sect. 6.3]. This is in line with the theory, see the second order optimality condition [27, Ch. 12.5]. Our experience confirms that for various OCPs, indefinite Hessians are crucial for fast local

convergence, especially for fine control grids. We can observe their behavior in the optimization by considering the iterates when they start being accepted. Here we employ the various optimal control examples outlined in Appendix B. The problems are discretized using 100 shooting and control intervals. We distinguish between problems based on the structure of their optimal solution.

### *4.1.1 Problems with sensitivity-seeking/singular arcs*

Intervals in which no bounds for states and controls are active and the optimal control is determined by time derivatives of the Hamiltonian are known as singular arcs, see e.g. [40]. Examples whose optimal solutions contain singular arcs include Catalyst mixing B.3, Egerstedt standard B.5, Electric car B.6, Goddard's rocket B.7 and Three tank multimode B.14. The Electric car problem in particular exhibits an iteration behavior typical for such problems.

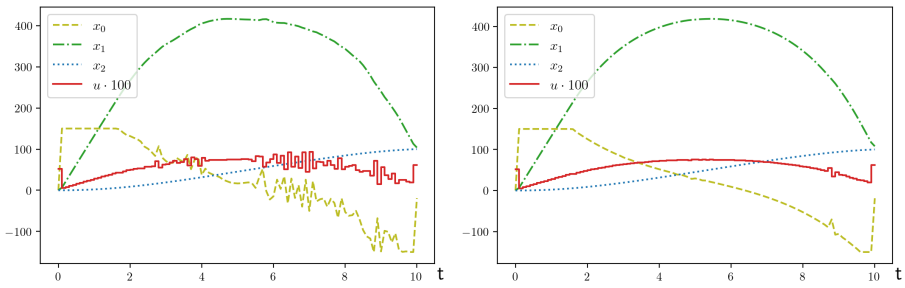



**Fig. 2** Iterates of the electric car problem before and after the first time the SR1 Hessian was accepted

The indefinite Hessian step smooths the controls on singular arcs while barely changing the states. This phenomenon can be explained either from the perspective of indirect methods, where time derivatives of the Hamiltonian determine the controls [40], or from the perspective of direct methods, where the controls are discretized. For a direct method, the fine control grid causes small directional derivatives of the state trajectory with respect to the controls. For demonstration, let $u_k^*, u_{k+1}^*, \ldots, u_J^* \in \mathbb{R}$ be optimal discretized controls on the grid $t_k, t_{k+1}, \ldots, t_J \subset [t_s, t_g]$, with neither states nor discretized controls being near their bounds. Adding oscillations, such as $\zeta := [\varepsilon, -\varepsilon, \ldots, \varepsilon]$ for $\varepsilon > 0$ to the controls to obtain $u_k + \varepsilon, u_{k+1} - \varepsilon, u_{k+2} + \mathbb{1} \cdot \varepsilon, \ldots, u_J + \mathbb{1} \cdot \varepsilon$ will in many cases barely change the states and a state dependent objective. This is best observed in an ODE where one control enters linearly

$$\dot{x}(t,u) = f(x(t,u)) + a \cdot u(t)$$

where $t \in [0,T]$, $a \neq 0$ is some vector and $f$ and its first derivatives are bounded. With an equidistant time grid $0 = t_0 < t_1 < t_N = T, \delta_t := t_k - t_{k-1} = \frac{T}{N}$, w.l.o.g. $N$ even, assume $u(t)$ is approximated piecewise constant by $u := u_0, u_1, \ldots, u_{N-1}$. With $\tilde{u} \leftarrow u_0 + \varepsilon, u_1 - \varepsilon, u_2, \ldots, u_{N-1}$, one can see with a Taylor approximation that

$$\begin{aligned} x(\delta_t, \tilde{u}) - x(\delta_t, u) &= a \cdot \varepsilon \delta_t + O(\varepsilon \delta_t^2) \\ x(2\delta_t, \tilde{u}) - x(2\delta_t, u) &= x(\delta_t, \tilde{u}) - x(\delta_t, u) - a \cdot \varepsilon \delta_t + O(\varepsilon \delta_t^2) = O(\varepsilon \delta_t^2) \end{aligned}$$

When $\tilde{u} \leftarrow u_0 + \varepsilon, u_1 - \varepsilon, u_2 + \varepsilon, \ldots, u_N - \varepsilon$, then using the above estimation $N/2$ times, one obtains

$$x(T,\tilde{u}) - x(T,u) = \frac{T}{2\delta_t} O(\varepsilon \delta_t^2) = O(\varepsilon \delta_t)$$

For small $\delta_t$, the final state barely changes although $\|\zeta\|_2$ may be quite large, because the dimension of $\zeta$ also increases proportionally to $\frac{1}{\delta_t}$. Determining the singular control requires good second derivative information, which the damped BFGS Hessian often does not provide sufficiently. Thus, eliminating oscillations such as this one can require many steps or even fail if the iterates do not reach a region where the SR1 Hessian is accepted. This can be observed - in particular - for the Catalyst mixing problem B.3.

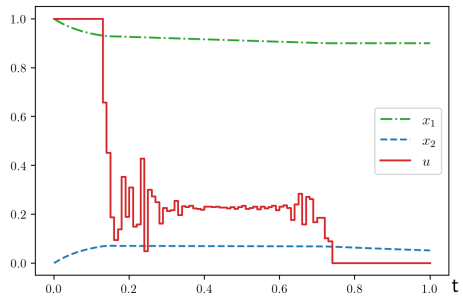


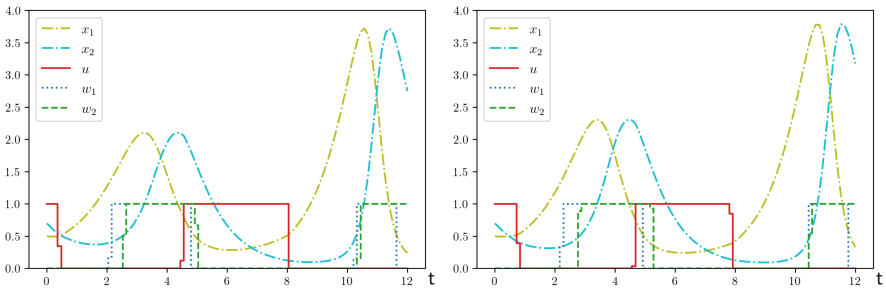

**Fig. 3** Iterate 100 of the catalyst mixing problem, the control $u$ still oscillates around the optimal trajectory

One reason for the poor performance of BFGS is - as mentioned in [23, Sect. 5.2ff] - that many damped BFGS updates are skipped owing to ill conditioning. This occurs especially once step norms become small when regions with oscillating controls are reached. When allowing more ill-conditioned updates, the problem remains as the damping factor $\theta$ often tends to zero in that case, meaning minor secant information is used to augment the Hessian. In Section 5.1, we discuss how poor scaling may play a role as well.

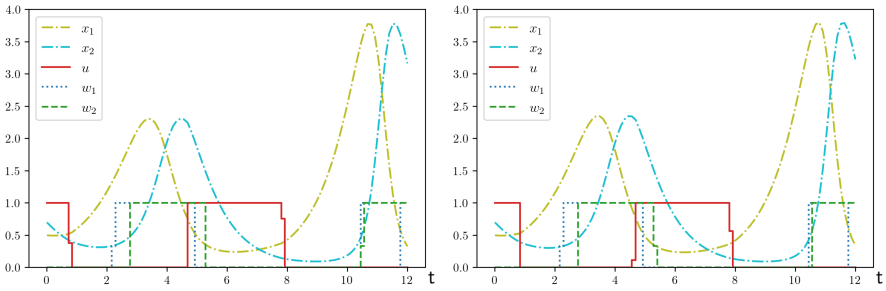
The point at which SR1 Hessians are accepted depends heavily on the specific path the iterations take; thus, it is very sensitive to algorithmic options, problem parameters, start points and used integrators. This complicates evaluation of the performance as small changes in the above settings can make the SR1 Hessian be accepted much sooner or later, significantly changing the number of required iterations. We account for this in Section 4.3 by testing for many slightly perturbed start points. One of our goals is to make the method more consistent.

### *4.1.2 Problems with bang-bang arcs*

Bang-bang arcs are intervals in which a control bound is active, compare [40]. If the control enters linearly, this is characterized by a switching function being nonzero. Most problems in the last section generated bang-bang arcs as well, as the controls $u$ had both upper and lower bounds. In this section, we consider problems whose optimal solution consists solely of bang-bang arcs and where the controls switch between their upper and lower bound. A problem class known to yield such solutions are optimum experimental design (OED) problems [37]. For the Lotka OED problem B.10 in particular, the following behavior can be observed.




**(a)** Iterate 15 **(b)** Iterate 80




**(c)** Iterate 105 **(d)** Iterate 106, SR1 Hessian was accepted

**Fig. 4** Iterates of the Lotka OED problem. The switching structure of the control and measurement functions is found within 15 iterations. The SR1 Hessian is required for fast convergence.

When the switching structure is found for the respective problem, the switching points must be shifted to the right times to reach the optimal solution. This requires changing discretized controls from lower - to upper bound and vice-versa. As with the oscillating controls in Sect. 4.1.1, this change is penalized by positive definite Hessians. Thus, the iterates tend to make slow progress until they reach a point where the SR1 Hessian is accepted, which can once again require a varying number of iterations.

## 4.2 Convexification strategies

Because the SR1 and exact Hessian have favorable convergence properties, regularizing these Hessians such that they are more often accepted, but retain most of their good properties, was suggested in [21, Sect. 7.1]. A QP-convexification strategy based on convex combinations between indefinite Hessian $H^{[1]}$ and fallback Hessian $H^{[2]}$ was already implemented. If $N_H > 2$ Hessians may be tried per SQP iteration, then convex combinations of the form

$$H \leftarrow (1 - \frac{k}{N_H - 1})H^{[1]} + \frac{k}{N_H - 1}H^{[2]} \tag{34}$$

for $k = 1, \ldots, N_H - 1$ are used to build a series of increasingly convexified QPs, which `qpOASES` attempts to solve in turn. While this strategy proved helpful on some of our problems, e.g. Goddard's rocket, it had minor effect on others, e.g. Catalyst mixing. The problem is that the regularized Hessians depend on the BFGS fallback Hessian, and may inherit its poor performance.

An alternative strategy taken from `ipopt` [43, 3.1] and previously tested for `blockSQP` [24] is regularization by adding scaled identities $I \cdot \kappa$. This strategy proved effective especially for the Hanging chain problem B.9. As the scaling factor $\kappa$ was increased whenever a convexified QP was rejected and reduced only at the beginning of each SQP iteration, it could become large if more than two Hessians were tried in each SQP iteration. This resulted in short steps from the regularized Hessian, thereby significantly increasing the number of required SQP iterations. We modified the scheme to instead adapt the scaling factor once at the end of each SQP iteration, based on which Hessian was accepted. Algorithm 1 below details our new strategy.

**Algorithm 1** : SQP loop with new convexification strategy

**Input:** Start point $\bar{x}^{(0)}$, algorithmic parameters
**Output:** Latest iterate $\bar{x}^{(*)}$ - local optimum in case of success

$H^{[1]}$ - first Hessian
$H^{[2]}$ - fallback Hessian
$N_H$ - maximum number of additional trial Hessians per SQP iteration ($\geq 2$)
$\kappa_0$ - initial scaling factor ($> 0$)
$\kappa_{max}$ - maximum scaling factor ($> 0$)
sol(QP) - QP solution procedure, may be **successful** and return a solution or fail
$\tau_H$ - tolerance factor ($> 0$)

do SQP initialization, initialize $H^{[1]}, H^{[2]}$
$\kappa \leftarrow \kappa_0$
**for each** SQP iteration $j = 1, \ldots$ **do**
    $k \leftarrow 0$
    $d_\xi \leftarrow 0$, $d_\lambda \leftarrow 0$

    **if successful** $d_\xi, d_\lambda \leftarrow$ sol(QP($H^{[1]}$)) **then** go to line 29 **else** $k \leftarrow k+1$
    **while** $k < N_H$ **do**
        $H \leftarrow H^{[1]} + I \cdot \kappa \cdot 2^{k-N_H+1}$
        **if successful** $d_\xi, d_\lambda \leftarrow$ sol(QP($H$)) **then** break **else** $k \leftarrow k+1$
    **end while**
    **if** $k > 1$ **and successful** $\tilde{d}_\xi, \tilde{d}_\lambda \leftarrow$ sol(QP($H^{[2]}$)) **then**
        **if** $k = N_H$ **or** $\|d_x\| \leq \tau_H \cdot \|\tilde{d}_x\|$ **then**
            $d_\xi \leftarrow \tilde{d}_\xi$, $d_\lambda \leftarrow \tilde{d}_\lambda$
        **end if**
    **end if**

    **if** $j > 1$ **and** $k > 0$ **and** $d_\xi \neq 0$ **then**
        **if** $N_H = 2$ **then**
            **if** $k = 2$ **then** $s_\kappa \leftarrow 1$ **else** $s_\kappa \leftarrow -1$
        **else**
            $s_\kappa \leftarrow k - (N_H - 1)$
            **if** $N_H > 3$ **and** $k = N_H$ **then** $s_\kappa \leftarrow s_k + (N_H - 3)$
        **end if**
        $\kappa \leftarrow \kappa \cdot 2^{s_\kappa}$
        **if** $\kappa > \kappa_{max}$ **then** $\kappa \leftarrow \kappa_{max}$
    **end if**

    **if** $d_\xi \neq 0$ **then**
        do filter line search with $d_\xi$
    **else**
        do QP error handling
    **end if**
    complete the SQP step, compute new Hessians $H^{[1]}$, $H^{[2]}$
**end for**

We recommend setting $N_H$ to 4 when using this strategy and chose this value in our experiments. It allows for quick adaptation of $\kappa$ while not requiring an excessive amount of QP solution attempts. $\kappa_{max}$ was set to default 8.0 and $\tau_H$ was set to $2/3$. In line 13, we disable the step norm condition whenever the first regularized Hessian is accepted. This often occurs when approaching a region where the first Hessian becomes acceptable and causes the scaling factor to rapidly decrease. Although the steps might be small at first, it often prevents the iterates from leaving that region and is helpful for consistent performance.
When using indefinite Hessians and especially when using regularized Hessians, we depend less on the BFGS Hessian and can allow it stronger positive definiteness, hence we increase the default damping factor in (13) from 0.2 to $\frac{1}{3}$. This tended to improve consistency in our experience and often resulted in faster QP solution.

For QPs in the form of (7) as arising from direct multiple shooting, we can derive another convexification strategy by applying Algorithm 1 after the condensing procedure from Section 3. In particular, the convexification strategy is applied to the condensed Hessian (17)

$$R + G^T Q G + S^T G + G^T S.$$

This is equivalent to regularizing $R$, i.e. adding the scaled identities only for indices corresponding to free variables. Thus, we can employ this alternative strategy independently of whether condensing is actually used, provided that free and dependent variables are known from each other. This new strategy is referred to as *convexification strategy 2* and Algorithm 1 as *convexification strategy 1*. We test both on different problems where the performance was poor or inconsistent before.

### 4.3 Numerical examples for the convexification strategy

We test our convexification strategies on our set of test problems. To eliminate the strong dependency on specific parameters and options through "happening upon a better path" we solve each problem for perturbed start points as specified in Appendix B. The complete results for Goddard's Rocket problem B.7 are shown in Fig. 5.

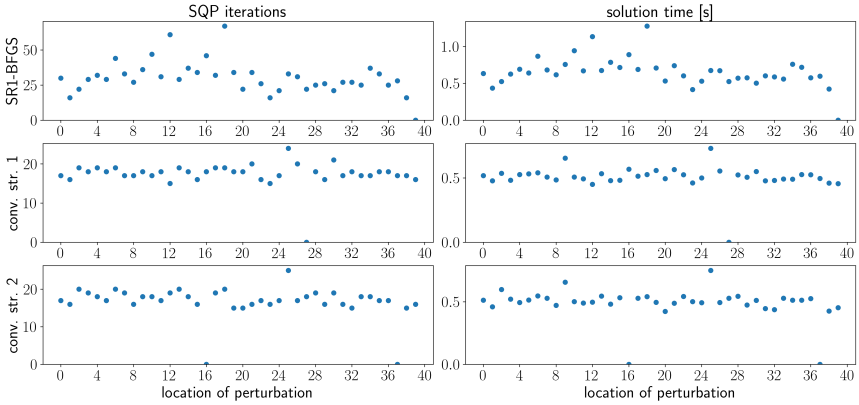



**Fig. 5** Total iterations and solution times for Goddard's rocket problem with perturbed start points for different Hessian sequences. A value of 0 indicates unsuccessful termination. Total iterations and solution times of unsuccessful attempts count towards the average

Despite the perturbations being small, the iteration count varies widely for SR1-BFGS. Regularized SR1 Hessians are accepted more reliably, lowering the number of required iterations and making it more consistent. Fig. 6 shows the averaged iteration counts and solution times for every example. The experiments were performed on an AMD Ryzen 5 3550H 4-core CPU running Ubuntu 24.04.3 with the scaling governor set to "performance".

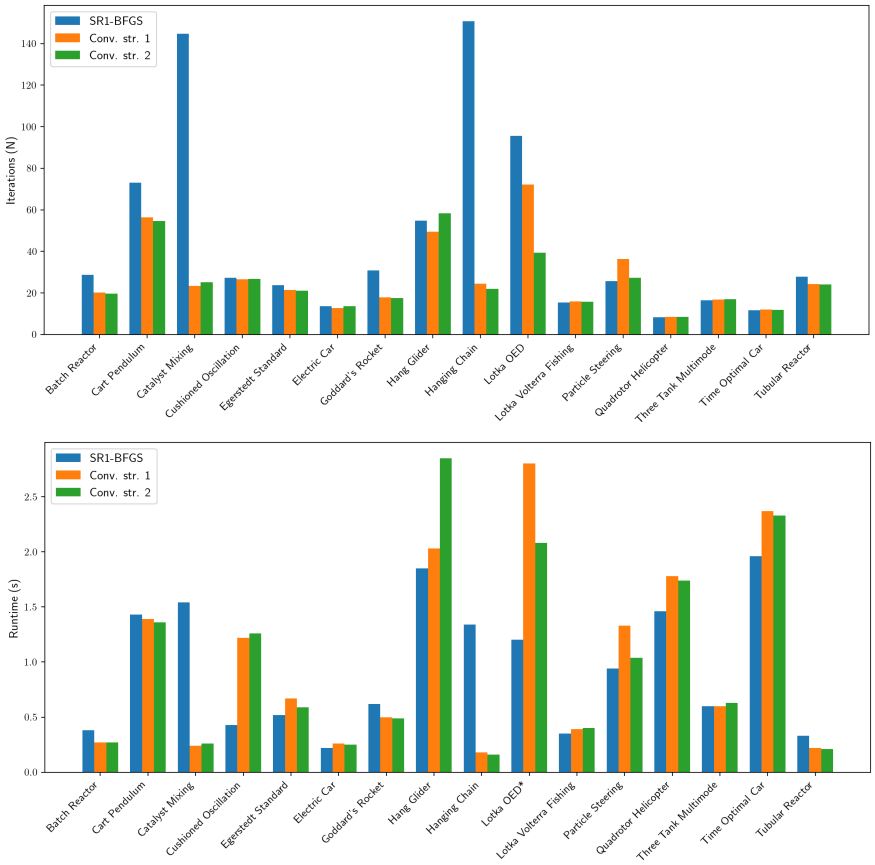



**Fig. 6** Iteration counts and solution times with and without convexification strategies, averaged over 40 runs with perturbed start points. The solver often failed for Catalyst mixing and Hanging chain under SR1-BFGS. The times for Lotka OED are scaled by 0.25

If the strategy significantly reduces the number of iterations, then the solution time is consequently reduced. Else the solution time may increase owing to the increased overhead of QPs being solved in sequence. For several problems, the performance and consistency improved, with two problems being solved reliably only when a convexification strategy is enabled. As convexification strategy 2 performed considerably better in two instances, we default to it for further numerical experiments.

As suggested in [21, Sect. 7.1], the QPs may be solved in parallel to reduce their overhead. An added benefit is that this allows terminating the QP solver if a QP takes long to yield a result. Such QPs typically occur if the regularization produces a Hessian that slightly violates the definiteness condition. We repeat the above experiments with parallel solution of QPs under convexification strategy 2.

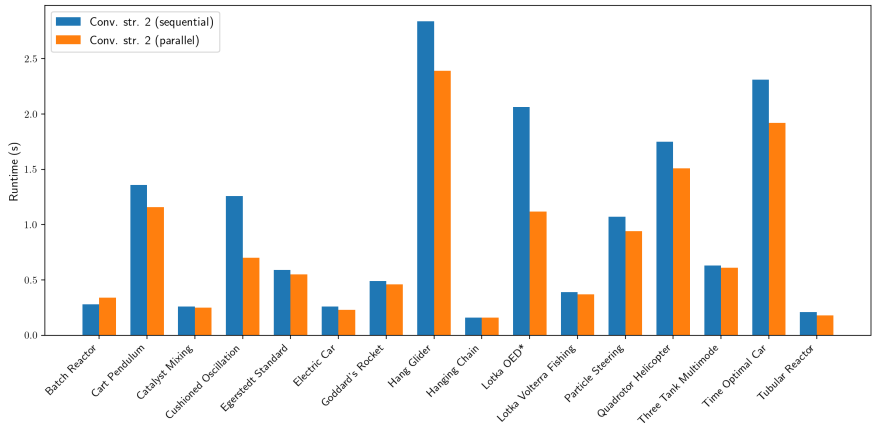



**Fig. 7** Solution times for sequential and parallel solution of QPs.
The times for Lotka OED are scaled by 0.25

The overhead was reduced for most problems. For Cushioned Oscillation and Lotka OED, the performance improved significantly owing to stalling QPs being terminated. A drawback is that this parallelization makes the solution time more sensitive to the computational environment, such as the scheduling behavior of the operating system.

## 5 Scaling and termination

In this section we consider the scale of problems and its connection to termination criteria, quasi-Newton updates and sizing strategies. We propose a termination method that is less sensitive to scale and a scaling heuristic to improve performance and consistency. Numerical tests show benefits and remaining shortcomings.

It is well known that the scale of the variables has a great effect on the performance of optimization algorithms [43, Sect. 3.8]. Newton's method is invariant to rescaling of variables $x \to S \cdot x$, $f(x) \to f(S^{-1}x)$ and functions $f(x) \to s_f \cdot f(x)$ where $S = diag(s_1, s_2, ..., s_n)$, $s_k > 0$, $s_f > 0$ and by extension one could expect optimization methods using the exact Hessian to be slightly sensitive to the scale of variables. As mentioned in [43, Sect. 3.8], scaling nevertheless often has a huge impact on the behavior of optimization algorithms. In our case, especially when much of the optimization relies on the BFGS Hessian, we expect an even greater impact. Thus, our aim is to improve performance by rescaling problems, especially for instances where the convexification strategies from the previous section performed poorly. The Lotka Volterra fishing problem B.11 is such an example and we discuss it in more detail.

### 5.1 Scaling case study

The Lotka Volterra fishing problem B.11 has the following optimal solution [35].

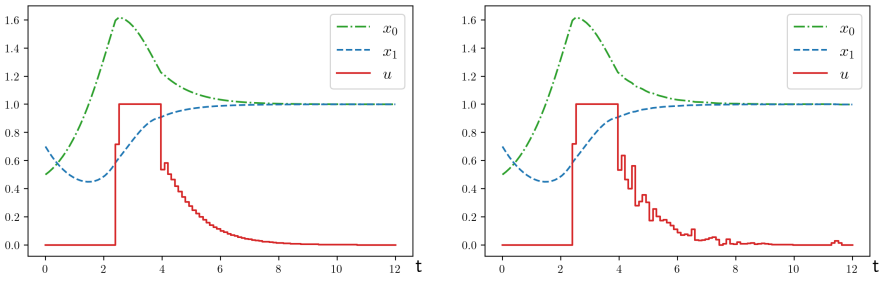
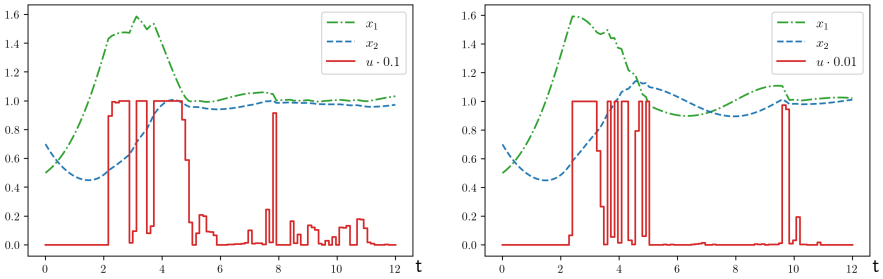



**Fig. 8** The optimal solution (left) and iterate 10 (right) of the Lotka Volterra fishing problem

Scaling $x_1$ by 100, $x_2$ by 0.01 and vice versa has a minor effect, resulting in 15 and 18 iterations, respectively. Scaling both by 100 and 0.01 results in 24 and 22 iterations, respectively. We used the default start point; the results were comparable for perturbed starts.

A stronger effect is achieved when rescaling the control $u$. Scaling by 10 and 100 results in 24 and 55 iterations, respectively. To explain this phenomenon, we observe the tenth iterate.




**(a)** Iterate 10, control scaled by 10.0

**(b)** Iterate 10, control scaled by 100.0

**Fig. 9** Tenth iterates of scaled Lotka Volterra fishing problems. Scaling the control higher results in stronger oscillations

As the control values are scaled higher, derivatives with respect to them are scaled lower. Second derivatives are affected twice, meaning any Hessian approximation must be scaled down significantly before large steps can be taken in $u$. For BFGS, a small step in some components tends to result in a small update in these components and poor downscaling. This phenomenon is known, and has been investigated by Powell [31] in a similar setting where the initial Hessian approximation is

$$H^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & \lambda \end{bmatrix}$$

for $\lambda \gg 1$, with the exact Hessian being the identity such that downscaling is required. An example closer to our setting would be the exact Hessian $H$ being

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

with

$$H^{(0)} = I,\ \delta = [1,\ \delta_2]^T,\ \gamma = [1,\ 0.1 \cdot \delta_2]^T \text{ for some } 0 < \delta_2 \leq 1.$$

Here, the SR1 update reproduces $H$ independently of $\delta_2$, while for $\delta_2 = 0.1$ the BFGS update yields

$$H^{(1)} \approx \begin{bmatrix} 1.008 & -0.08 \\ -0.08 & 0.99 \end{bmatrix}$$

i.e. almost no downscaling occurs in the second variable.

For our optimal control example, steps in discretized control variables, where steps were initially small, tend to again be small. This causes few control variables to change considerably, which results in the strong oscillations at iterate 10. Considering the opposite direction and scaling $u$ by 0.1 instead yields the optimal solution at iterate 10.

Our goal when rescaling is to ensure good quasi-Newton BFGS updates with respect to the free variables, as optimal free variables guarantee optimal dependent variables when feasibility holds. At this point, we consider the sizing strategies, as their aim is supporting the quasi-Newton updates as well. We review the termination criteria, as they are heavily influenced by the scale of problems.

### 5.2 Sizing strategies, scaling, termination criteria

The sizing strategies [9,28] account for the objective - and variable scale of problems. They scale the Hessian approximation such that its convex spectrum overlaps with that of the exact Hessian. Thus, they adapt to the scale of all variables and the objective function, but do not consider the scale of variables relative to each other. Like quasi-Newton updates, they are applied block-wise. The scale of the constraints does not need to be considered in our setting as they are already scaled by the Lagrange multipliers. The aim of our scaling is therefore both assuming scaling effort off of the quasi-Newton updates and improving the updates for free components.

Related to the scale of the objective and variables are the termination criteria. The original implementation of `blockSQP` terminates successfully if the optimality error $\|\nabla L(x,\lambda)\|_\infty$ and constraint violation $\|c(x)\|_\infty$ satisfy [23, Sect. 1.1]

$$\frac{\|\nabla L(x,\lambda)\|_\infty}{1 + \|\lambda\|_\infty} < \varepsilon_{opt},\ \|c(x)\|_\infty \leq \varepsilon_{feas}. \tag{35}$$

Note that the constraints are formulated as $c(x) \geq 0$ and the Lagrangian is written as $L(x,\lambda) = f(x) + \lambda^T c(x)$, which we preserve for this section.

Termination depends directly on the scale of the constraint functions, the scale of the objective function and the scale of the variables. The acceptable constraint violation should be considered to be best set during modeling and OCP formulation stages. To account for scale, allowing different tolerances for different constraints should be considered for the future. For the optimality, it remains unclear how $\varepsilon_{opt}$ should be chosen a-priori for a particular

problem to obtain a suitably accurate solution. It depends on the scale of both the objective function and the variables, which complicates determining a suitable reference scale. Rather than relying on additional problem knowledge, we instead opt to consider the progress made for deciding when to terminate, as it is already being monitored via the filter line search.

## 5.3 Termination criteria

Our goal is to augment the termination criteria to allow both for premature termination if a solution of acceptable accuracy has been found and extra steps for increased accuracy. The former feature is already available in many NLP solvers such as `ipopt` [43, 3.1], where it is possible to specify an *acceptable tolerance* $\varepsilon_{acc} > \varepsilon_{opt}$.

The criteria (35) take into account that the Lagrange multipliers may become large if the active constraint gradients are nearly linearly dependent [43]. Depending on the problem, the default tolerances $\varepsilon_{opt}$, $\varepsilon_{feas} = 10^{-6}$ may be too small to be achievable or too large to yield an accurate solution. For example, the three tank multimode problem B.14 requires $\varepsilon_{opt} \leq 10^{-9}$ to yield a satisfactory solution.

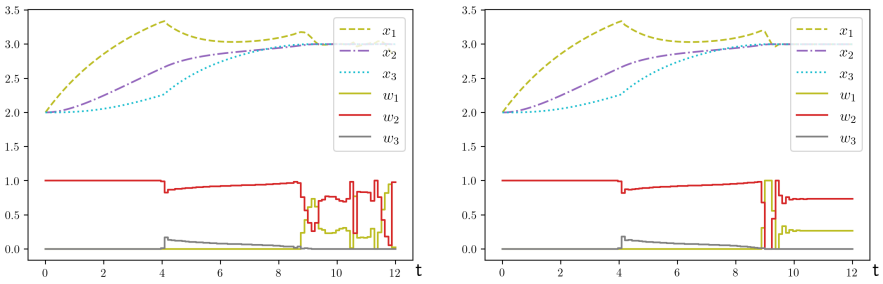



**Fig. 10** Optimal solutions of the three tank multimode problem with different accuracy. Left is a solution with an optimality error of $4.92 \cdot 10^{-7}$, right is an accurate solution with an optimality error below $10^{-9}$

On the other hand, the default tolerance of $\varepsilon_{opt} = 10^{-6}$ may be too small to achieve for some problems due to numerical inaccuracies. Such problems include the Electric car problem B.6 and Goddard's rocket problem, causing unsuccessful termination for some start points, as seen previously in Fig. 5 in Section 4.3.

### *5.3.1 Line search heuristics and flexible termination criteria*

To prevent the issues above and enable flexible termination, we modify the termination criteria to consider the progress made. In our setting, progress is ensured through the filter conditions, the line search and certain heuristics for when the line search fails. Hence, we embed our new termination scheme into these existing procedures. In particular, `blockSQP` employs the following whenever the line search fails [21, Sect. 6.2].

a) Check if accepting the full step reduces the KKT error by a factor $\kappa_F$ in the sense that

$$\max\left\{\|c(x^{(j+1)})\|_\infty, \frac{\|\nabla_x L(x^{(j)},\lambda^{(j+1)})\|_\infty}{1+\|\lambda^{(j+1)}\|_\infty}\right\} \leq \kappa_F \cdot \max\left\{\|c(x^{(j)})\|_\infty, \frac{\|\nabla_x L(x^{(j)},\lambda^{(j)})\|_\infty}{1+\|\lambda^{(j)}\|_\infty}\right\}. \tag{36}$$

In that case, accept the step. We refer to this as the *KKT heuristic*.

b) Attempt to reduce the constraint violation and find a point acceptable for the filter through a heuristic. In case of multiple shooting, one can set the stage states through integration over the full time horizon [21, Sect. 6.3.2] to fulfill the continuity conditions (3). We refer to this as the *feasibility heuristic*.

c) Recompute a step with the identity Hessian and reattempt the line search.

d) If the constraint violation is above a tolerance, invoke feasibility restoration.

- terminate unsuccessfully

To enable flexible termination, we compare and possibly save iterates that fulfill

$$\|c(x)\|_\infty \leq \varepsilon_{feas}, \frac{\|\nabla_x L(x,\lambda)\|_\infty}{1+\|\lambda\|_\infty} \leq \varepsilon_{acc} \tag{37}$$

for some acceptable tolerance $\varepsilon_{acc} > \varepsilon_{opt}$. This enables us to return an acceptable iterate as a fallback in case the line search fails. Similarly, we add the option to perform up to $N_{ref} > 0$ steps for improved accuracy, saving iterates which fulfill the termination criteria (35). If the line search fails afterwards or the maximum number of additional steps has been reached, we return the best iterate in terms of optimality error and constraint violation.

In addition, we made modifications and additions to the line search error handling. In the KKT heuristic, it is desirable to have the condition

$$\frac{\|\nabla_x L(x^{(j+1)},\lambda^{(j+1)})\|_\infty}{1+\|\lambda^{(j+1)}\|_\infty} \leq \kappa_F \cdot \frac{\|\nabla_x L(x^{(j)},\lambda^{(j)})\|_\infty}{1+\|\lambda^{(j)}\|_\infty}. \tag{38}$$

As this requires evaluating first derivatives, we check this condition whenever a step was accepted through 36 once first derivatives are available. If (38) does not hold, we disable the KKT heuristic until a line search was successful. This prevents issues with the KKT heuristic sometimes being invoked repeatedly for iterates that make little progress.

Due to numerical inaccuracies or old information, it may sometimes be beneficial for steps to ignore the filter, compare [43]. In our experience, such problems occurred mostly in regions of local convergence. We therefore opt for allowing steps to ignore the filter up to a limited number of times if the current iterate is within acceptable tolerance after the filter line search has failed. This often allows the linesearch to succeed again and find an optimal solution, as opposed to one that is merely acceptable.

### *5.3.2 Combining filter line search - and termination procedures*

We need to embed the changes and additions discussed in the previous section into the existing line search - and termination procedures. The main question is in what order the line search error handling heuristics should be performed. We opt for the following, attempting each fallback in turn till one succeeds.

a) If a solution was previously saved in step j), return it and declare *success*.

b) Attempt the KKT heuristic.
c) If the used Hessian is not the fallback Hessian, recompute the step with the fallback Hessian and reattempt the line search.
d) If $\frac{\|\nabla_x L(x,\lambda)\|_\infty}{1+\|\lambda\|_\infty} \leq \varepsilon_{acc}$ and $\|c(x)\|_\infty \leq \varepsilon_{feas}$, take the step and overwrite dominating elements of the filter. Allow this only a limited number of times.
e) If an iterate was previously saved in step i), return it and declare *partial success*.
f) Invoke the feasibility heuristic.
g) Recompute the step with the identity Hessian, reattempt the line search.
h) Invoke feasibility restoration if the constraint violation is above a tolerance.
- terminate unsuccessfully.

If the filter line search or a fallback succeeded, perform the following.

i) If $\frac{\|\nabla_x L(x,\lambda)\|_\infty}{1+\|\lambda\|_\infty} \leq \varepsilon_{acc}$ and $\|c(x)\|_\infty \leq \varepsilon_{feas}$, save the iterate.
j) Check if the KKT error is below the tolerance for successful termination. If yes, terminate or save the iterate and perform additional steps for improved accuracy.
k) If the maximum number of additional steps $N_{acc}$ has been performed, return the best saved iterate.

By default, we allow the filter to be overwritten two times. Additional steps for increased accuracy must be explicitly enabled. The accurate solution of the three tank multimode problem B.14 shown in Fig. 10 was obtained by allowing up to 10 additional steps. The electric car problem B.6 is now solved with partial success for two start points, avoiding convergence failure.

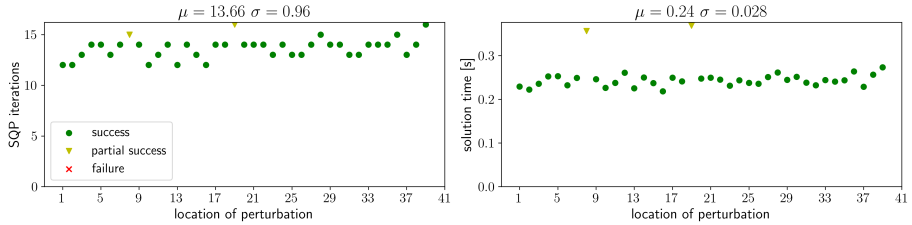



**Fig. 11** Total iterations and solution times of the electric car problem for perturbed start points. Allowing premature termination avoided convergence failure two times, causing partial success instead

The linesearch heuristics incur some computational overhead, but few additional iterations are performed before partial success is declared.

## 5.4 An automatic scaling heuristic

As discussed above, we develop a rescaling that improves the quasi-Newton updates, and compute the scaling using the same information -

$$d^{(*)} = x^{(k)} - x^{(k-1)}, \quad \gamma^{(*)} = \nabla L(x^{(k)}, \lambda^{(k)}) - \nabla L(x^{(k-1)}, \lambda^{(k)}),$$

where we distinguish between dependent and free components $x$, $u$ -

$$d^{(*)} = d_x^{(*)}, d_u^{(*)}, \quad \gamma^{(*)} = \gamma_x^{(*)}, \gamma_u^{(*)}.$$

We compute a rescaling factor for $u$ as follows.

**Algorithm 2** : Scaling heuristic for free variables

**Input parameter:** $N > 0$ number of available past $d$ - $\gamma$ pairs
**Input data:** $d^{(1)}, ..., d^{(N)}$ - $N$ previous steps
$\gamma^{(1)}, ..., \gamma^{(N)}$ - $N$ previous Lagrange gradient differences
**Output:** $S_u > 0$ rescaling factor for free variables

$n_x, n_u$ - number of dependent and free variables
$\varepsilon_s, \varepsilon_m > 0$ - nonzero tolerances for step components and means, set to $10^{-8}$, $5 \cdot 10^{-7}$
*mean* - arithmetic mean over a set with $mean\{\} = 0$

*Compute ratios $r^{(d)}$, $r^{(\gamma)}$ of free to dependent components*
$\bar{d}_x \leftarrow 0$, $\bar{d}_u \leftarrow 0$, $\bar{\gamma}_x \leftarrow 0$, $\bar{\gamma}_u \leftarrow 0$
$r^{(d)} \leftarrow 1.0$, $r^{(\gamma)} \leftarrow 1.0$, $c_d \leftarrow 0$, $c_\gamma \leftarrow 0$
**for** $k = 1, ..., N$ **do**
    $\bar{d}_x \leftarrow mean\{|d^{(k)}_{x,j}|$ **for** $j = 1, ..., n_x$ **if** $|d^{(k)}_{x,j}| \geq \varepsilon_s\}$
    $\bar{\gamma}_x \leftarrow mean\{|\gamma^{(k)}_{x,j}|$ **for** $j = 1, ..., n_x$ **if** $|d^{(k)}_{x,j}| \geq \varepsilon_s\}$
    $\bar{d}_u \leftarrow mean\{|d^{(k)}_{u,j}|$ **for** $j = 1, ..., n_u$ **if** $|d^{(k)}_{u,j}| \geq \varepsilon_s\}$
    $\bar{\gamma}_u \leftarrow mean\{|\gamma^{(k)}_{u,j}|$ **for** $j = 1, ..., n_u$ **if** $|d^{(k)}_{u,j}| \geq \varepsilon_s\}$

*Use geometric mean for ratios*
    **if** $\bar{\gamma}_u > \varepsilon_m \wedge \bar{\gamma}_x > \varepsilon_m$ **then**
        $r^{(\gamma)} \leftarrow r^{(\gamma)} \cdot (\bar{\gamma}_u / \bar{\gamma}_x)$
        $c_\gamma \leftarrow c_\gamma + 1$
        **if** $\bar{d}_u > \varepsilon_m \wedge \bar{d}_x > \varepsilon_m$ **then**
            $r^{(d)} \leftarrow r^{(d)} \cdot (\bar{d}_u / \bar{d}_x)$
            $c_d \leftarrow c_d + 1$
        **end if**
    **end if**
**end for**

**if** $c_d > 0$ **then** $r^{(d)} \leftarrow (r^{(d)})^{1/c_d}$ **else** $r^{(d)} \leftarrow 1.0$
**if** $c_\gamma > 0$ **then** $r^{(\gamma)} \leftarrow (r^{(\gamma)})^{1/c_\gamma}$ **else** $r^{(\gamma)} \leftarrow 1.0$

*Heuristic to compute the scaling factor*
$S_u \leftarrow 1.0$
**if** $r^{(\gamma)} > 10.0$ **then**
    $S_u \leftarrow r^{(\gamma)} / 10.0$
**else if** $r^{(\gamma)} < 1.0$ **then**
    **if** $r^{(d)} > 1.0$ **then**
        **if** $r^{(\gamma)} < 0.1$ **then**
            $S_u \leftarrow r^{(\gamma)} \cdot 10.0$
        **else**
            $S_u \leftarrow \min\{r^{(\gamma)} \cdot r^{(d)}, 1.0\}$
        **end if**
    **else**
        $S_u \leftarrow r^{(\gamma)}$
    **end if**
**end if**

**return** $S_u$

For the Lagrange gradient differences, we aim to maintain the the accumulated free-dependent ratio $r^{(\gamma)}$ between 1 and 10. At the end of the optimization, this ratio often becomes small while the corresponding step ratio becomes large, compare Section 4.1.1. In this case, we allow for $r^{(\gamma)}$ to decrease down to 0.1, a threshold we found to work well.

We compute and apply the scaling factor $S_u$ in steps 1-3, invoking Algorithm 2 with $N =$ 1-3 and every fifth total iteration afterwards with $N = 5$. Whenever feasibility restoration is invoked, we discard all previous $d$ - $\gamma$ pairs. Thereafter, Algorithm 2 is invoked with the available past steps, up to $N = 5$.

## 5.5 Numerical experiments for the scaling heuristic

We apply our scaling heuristic in addition to the parallel version of convexification strategy 2 from Section 4.2. For problems that are already well scaled, we do not expect any performance benefits from our scaling. In this case, we demand that it does not noticeably impair the performance. Fig. 12 shows the results for the test problems.

The scaling strategy performs well for Lotka Volterra fishing and Hang glider while introducing minor overhead for some problems. Particle steering significantly improved in solution time as the QP solutions were sped up.

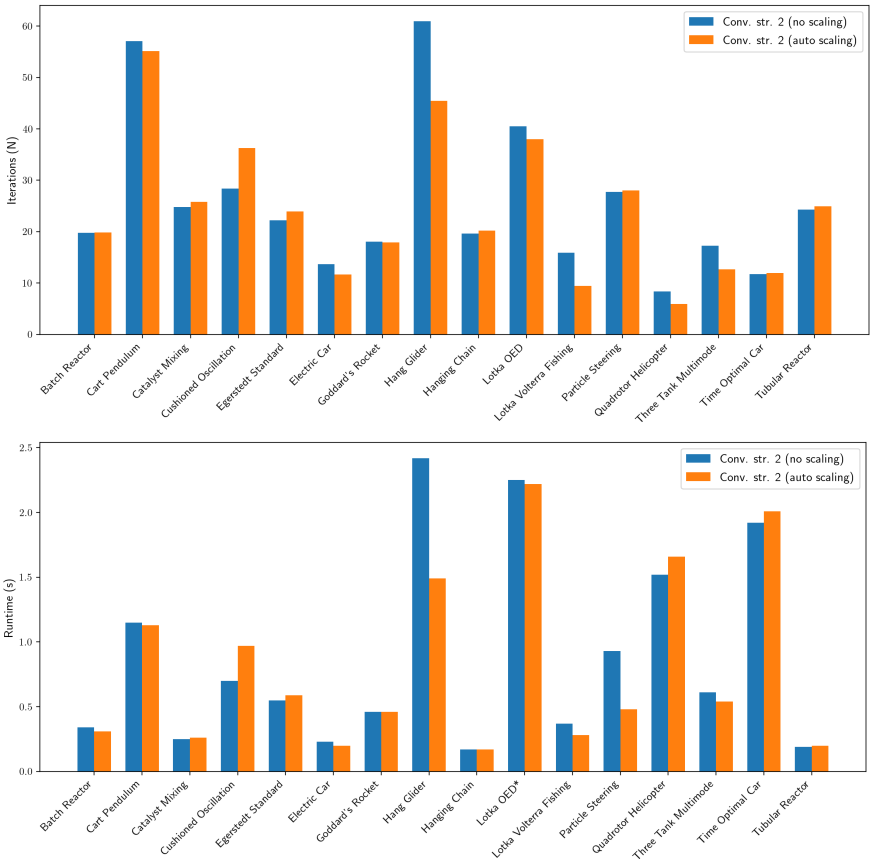



**Fig. 12** Iteration counts and solution times with and without the scaling heuristic
The times for Lotka OED are scaled by 0.5

The performance was poor for Cushioned oscillation. The problem is that right now, we scale all free variables equally and not relative to each other. These variables include the

stage-localized time scaling parameter $\delta_t$, which scales the ODE $\dot{x} = \delta_t \cdot f(x)$ such that the problem is reformulated as an equivalent problem with constant time horizon $[0, 1]$ for each shooting stage. As we have 100 shooting stages, each stage time parameter is effectively scaled by $\frac{1}{100}$. If we upscale this parameter by 100, the performance significantly improves and is no longer impaired by our scaling heuristic. Further upscaling the parameter by 5 improves the performance even more.

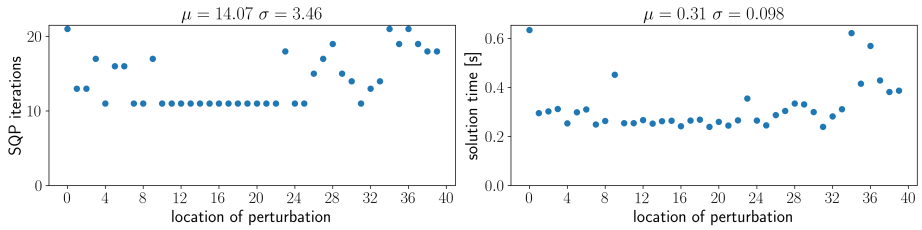



**Fig. 13** Total iterations and solution times for the cushioned oscillation problem B.4 for perturbed start points. The duration parameter was manually scaled by 500.0. Convexification strategy 2 and automatic scaling were enabled

This encourages investigation of more sophisticated scaling heuristics in the future, which consider the above issue. Thereby, an even wider range of problems may be covered and the performance improved.

## 6 Conclusions and outlook

In this section, we conclude this study with a comparison of our updated version of `blockSQP` to the original one and `ipopt`. From now on, we refer to our version as `blockSQP 2`. `blockSQP 1` now refers to the original version released by Dennis Janka [23] which our version is based on. It is not to be confused with the version available through `CasADi` [3], which is a separate implementation employing `CasADi` data structures and functions.

We discuss remaining shortcomings and future improvements.

### 6.1 Comparison of `ipopt`, `blockSQP 1` and `blockSQP 2`

We are interested in how `blockSQP 2` fares against the well established NLP solver `ipopt` [43] on our set of test problems. We made the following observations.

- `blockSQP 2` often converges faster locally, achieving higher accuracy in fewer steps and enabling efficient adaptive termination as established in Section 5.3.
- `ipopt` often requires more iterations, but has lower computational cost per iteration.
- `ipopt` itself requires only one thread to run, but benefits more from sensitivity generation being parallelized over the shooting intervals.
- `ipopt` tends to benefit more from the exact Hessian than `blockSQP`.

A fair comparison is therefore complicated. We opt to perform experiments for both quasi-Newton and the exact Hessian and with an optimality tolerance of $10^{-6}$. Parallel integration is enabled to allow both to utilize the 4 available CPU cores. `blockSQP 2` is run with convexification strategy 2 and automatic scaling, but no termination with partial success or steps for improved accuracy. For the comparison to the old version, we invoke `blockSQP 1` for

SR1-BFGS, enabling the original convexification strategy with $N_H = 4$ as described at the start of Section 4.2. Without the convexification strategy, `blockSQP 1` often fails on Lotka OED, with the performance being better for some problems and worse for others.

Detailed results are contained in Table 4 in the Appendix. Fig. 14 below shows the runtime comparison for exact Hessian `ipopt`, quasi-Newton `blockSQP 2`, and `blockSQP 1`.

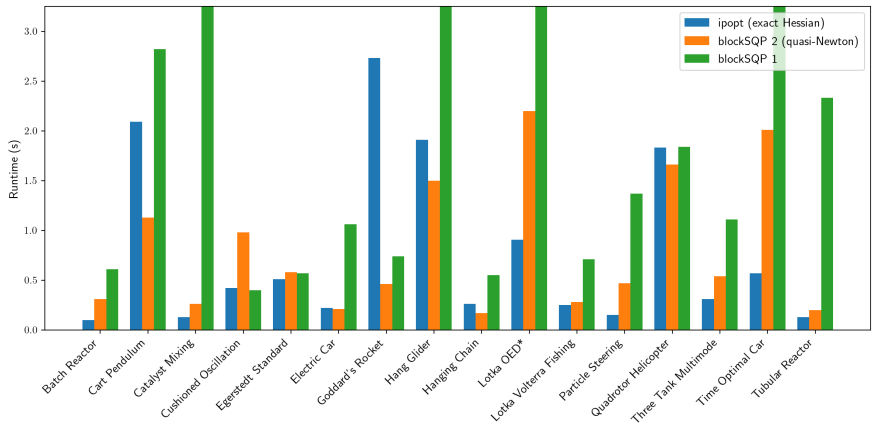



**Fig. 14** Solution times for the NLP solvers `ipopt`, `blockSQP` and `original blockSQP` for problems using a fixed step explicit Runge-Kutta-4 ODE integrator.
The times for Lotka OED are scaled by 0.5

`blockSQP 2` outperforms `blockSQP 1` on all problems except Cushioned oscillation. On the other hand, `ipopt` using the exact Hessian solves all problems reliably and outperforms `blockSQP 2` on most. It is important to note that a two-step explicit Runge-Kutta-4 scheme was used for integration over each shooting interval. This makes evaluating the exact Hessian computationally cheap. The benefit of quasi-Newton updates becomes significant if a more expensive scheme is used. We repeat the above experiment for `ipopt` and `blockSQP 2` using a two-step implicit scheme. The results are shown in Figure 15 and Table 5 in the Appendix.

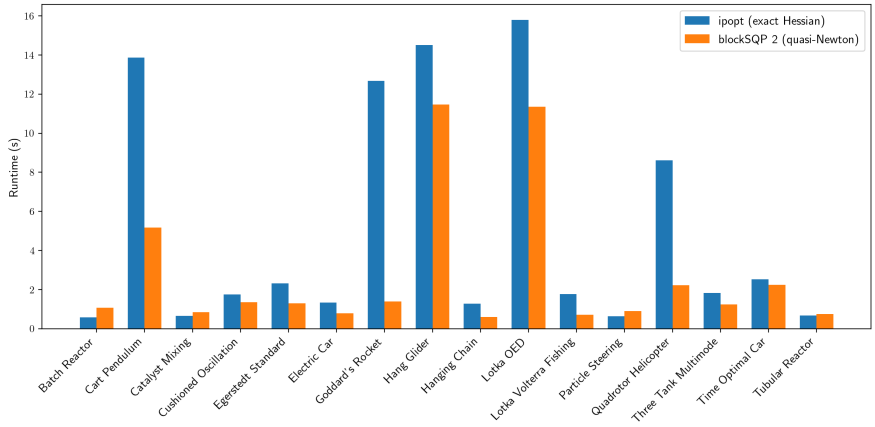



**Fig. 15** Solution times for the NLP solvers `ipopt` and `blockSQP` for problems using a fixed step implicit ODE integrator

In this scenario, quasi-Newton `blockSQP 2` outperforms exact Hessian `ipopt` on most problems. If an adaptive integrator is used, the computational cost of sensitivity generation may increase further. As adaptive ODE integration typically causes discontinuous integration errors, it may prevent convergence if the error tolerance is set too high. This affects `blockSQP 2` more than `ipopt` as the block-wise quasi-Newton updates quickly accumulate integration errors. Hence, it is difficult to make a fair comparison between the solvers in this scenario. These issues can be overcome by employing *Internal Numerical Differentiation* [5,7] inside the adaptive integrator. If this method is not available, an alternative is to rely on the termination scheme from Section 5.3.2 to stop the iterations and possibly declare partial success if integration errors prevent the solver from achieving the desired accuracy.

## 6.2 Summary and future work

Structured nonlinear programs arise in mathematical optimal control via direct multiple shooting. This structure information can be employed in sequential quadratic programming. We established the concept of implicit bounds in the context of dynamic optimal control and discussed how to implement them during modeling and discretization of optimal control problems. The convexification strategies and scaling heuristic can serve as a base for more sophisticated future methods and implementations. Extending the scaling heuristic such that it is beneficial for a wider range of problems can be achieved by scaling free variables individually or by treating variables corresponding to parameters in a separate way.

In the above work, we only made changes inside the optimization algorithm. Outside of it, dynamic adaptations to the problem formulation once a solution structure becomes apparent may be of interest as well. An alternative method to manage oscillating controls on singular arcs could be adding a regularization term, such as a Tikhonov regularization

$$\theta \cdot \sum_{k=1}^{N} \|u_k - u_{k-1}\|^2$$

or similar for the objective, although this could lead to a suboptimal solution and not preserve the block structure. One can monitor the iterates and smooth the controls if oscillations are detected. For problems with bang-bang arcs, switching time optimization [36] may be of interest once the iterates show switching behavior. This requires problem reformulation and solver reinitialization, but could improve convergence of the switching points to the optima and prevent issues arising from discretization points not aligning with switching points. Here, it is important to determine how algorithmic data, such as Hessian approximations and sizing information, may be adapted to the new problem to warmstart its optimization.

**Acknowledgements** We would like to thank Editage (www.editage.com) for English language editing.

**Funding** This project has received funding from the European Regional Development Fund (grants timingMatters and IntelAlgen) under the European Union's Horizon Europe Research and Innovation Program, from the research initiative "SmartProSys: Intelligent Process Systems for the Sustainable Production of Chemicals" funded by the Ministry for Science, Energy, Climate Protection and the Environment of the State of Saxony-Anhalt, and from the German Research Foundation DFG within GRK 2297 'Mathematical Complexity Reduction' and priority program 2331 'Machine Learning in Chemical Engineering' under grant SA 2016/3-1, which we gratefully acknowledge.

**Code and Data Availability** The full code and data were made available for review. We remark that a set of packages were used in this study, that were either open source or available for academic use. Specific references are included in the reference section.

## Declarations

**Competing interests** The authors have no relevant financial or non-financial interests to disclose.

## References


1. Amestoy, P., Duff, I.S., Koster, J., L'Excellent, J.Y.: A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. SIAM Journal on Matrix Analysis and Applications **23**(1), 15–41 (2001)
2. Andersson, J.A., Frasch, J.V., Vukov, M., Diehl, M.: A Condensing Algorithm for Nonlinear MPC with a Quadratic Runtime in Horizon Length. Automatica pp. 97–100 (2013)
3. Andersson, J.A., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M.: CasADi: a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation **11**(1), 1–36 (2019)
4. Beal, L.D., Hill, D.C., Martin, R.A., Hedengren, J.D.: Gekko optimization suite. Processes **6**(8), 106 (2018)
5. Bock, H.: Recent Advances in Parameter Identification Techniques for O.D.E. In: P. Deuflhard, E. Hairer (eds.) Numerical Treatment of Inverse Problems in Differential and Integral Equations, pp. 95–121. Birkhäuser, Boston (1983)
6. Bock, H., Plitt, K.: A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. In: Proceedings of the 9th IFAC World Congress, pp. 242–247. Pergamon Press, Budapest (1984)
7. Bock, H.G.: Internal Numerical Differentiation Revisited: Differentiation of Adaptive Integrators for ODE and DAE, Including Discontinuous Models. Second Argonne Theory Institute on Differentiation of Computational Approximations to Functions p. 4 (1998)
8. Conn, A.R., Gould, N.I., Toint, P.L.: Trust-Region Methods. SIAM (2000)
9. Contreras, M., Tapia, R.: Sizing the BFGS and DFP Updates: Numerical Study. Journal of Optimization Theory and Applications **78**(1), 93–108 (1993)
10. Dolan, E., Moré, J., Munson, T.: Benchmarking Optimization Software with COPS 3.0. Tech. Rep. ANL/MCS-TM-273, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, U.S.A. (2004)
11. Egerstedt, M., Wardi, Y., Axelsson, H.: Transition-time optimization for switched-mode dynamical systems. IEEE Transactions on Automatic Control **51**, 110–115 (2006)
12. Ferreau, H.: `qpOASES` – An Open-Source Implementation of the Online Active Set Strategy for Fast Model Predictive Control. In: Proceedings of the Workshop on Nonlinear Model Based Control – Software and Applications, Loughborough (2007)
13. Fletcher, R.: A new approach to variable metric algorithms. The Computer Journal **13**, 317–322 (1970). DOI 10.1093/comjnl/13.3.317
14. Garmatter, D., Maggi, A., Wenzel, M., Monem, S., Hahn, M., Stoll, M., Sager, S., Benner, P., Sundmacher, K.: Power-to-Chemicals: A Superstructure Problem for Sustainable Syngas Production. In: Mathematical Modeling, Simulation and Optimization for Power Engineering and Management, pp. 145–168. Springer (2021)
15. Gill, P.E., Murray, W., Wright, M.H.: Practical reduction methods for linear constraints. Communications of the ACM **24**(5), 296–306 (1981)
16. Gillula, J.H., Hoffmann, G.M., Huang, H., Vitus, M.P., Tomlin, C.J.: Applications of hybrid reachability analysis to robotic aerial vehicles. The International Journal of Robotics Research **30**(3), 335–354 (2011)
17. Goddard, R.H.: A method of reaching extreme altitudes (with 10 plates), vol. 2540. Smithsonian institution (1921)
18. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020). DOI 10.1038/s41586-020-2649-2

19. Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). DOI 10.1109/MCSE.2007.55
20. Jakob, W., Rhinelander, J., Moldovan, D.: pybind11 – Seamless operability between C++11 and Python (2017). https://github.com/pybind/pybind11
21. Janka, D.: Sequential quadratic programming with indefinite Hessian approximations for nonlinear optimum experimental design for parameter estimation in differential-algebraic equations. Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg (2015)
22. Janka, D.: Reference version of blockSQP archived by R. Wittmann (2025). DOI 10.5281/zenodo.17400197. URL `https://doi.org/10.5281/zenodo.17400197`
23. Janka, D., Kirches, C., Sager, S., Wächter, A.: An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal Hessian matrix. Mathematical Progamming Computation **8**(4), 435–459 (2016). DOI 10.1007/s12532-016-0101-2
24. Le, D.D.: Fork of CasADi. `https://github.com/doducle/casadi/tree/develop` (2017)
25. Leineweber, D., Bauer, I., Schäfer, A., Bock, H., Schlöder, J.: An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II). Computers & Chemical Engineering **27**, 157–174 (2003)
26. Logsdon, J., Biegler, L.: Decomposition strategies for large-scale dynamic optimization problems. Chemical Engineering Science **47**(4), 851–864 (1992)
27. Nocedal, J., Wright, S.: Numerical Optimization, first edn. Springer Verlag, Berlin Heidelberg New York (1999). ISBN 0-387-98793-2 (hardcover)
28. Oren, S.S., Luenberger, D.G.: Self-Scaling Variable Metric (SSVM) Algorithms: Part I: Criteria and Sufficient Conditions for Scaling a Class of algorithms. Management Science **20**(5), 845–862 (1974)
29. Powell, M.: Algorithms for nonlinear constraints that use Lagrangian functions. Mathematical Programming **14**(3), 224–248 (1978)
30. Powell, M.: The convergence of variable metric methods for nonlinearly constrained optimization calculations. In: O. Mangasarian, R. Meyer, S. Robinson (eds.) Nonlinear Programming 3. Academic Press (1978)
31. Powell, M.: How bad are the BFGS and DFP methods when the objective function is quadratic? Mathematical Programming **34**, 34–47 (1986)
32. Robinson, S.M.: A quadratically-convergent algorithm for general nonlinear programming problems. Mathematical Programming **3**(1), 145–156 (1972)
33. Roose, A.I., Yahya, S., Al-Rizzo, H.: Fuzzy-logic control of an inverted pendulum on a cart. Computers & Electrical Engineering **61**, 31–47 (2017)
34. Rutquist, P.E., Edvall, M.M.: PROPT - Matlab Optimal Control Software. Tomlab Optimization Inc (2010)
35. Sager, S.: Numerical methods for mixed–integer optimal control problems. Der andere Verlag, Tönning, Lübeck, Marburg (2005). ISBN 3-89959-416-9
36. Sager, S.: Reformulations and Algorithms for the Optimization of Switching Decisions in Nonlinear Optimal Control. Journal of Process Control **19**(8), 1238–1247 (2009)
37. Sager, S.: Sampling Decisions in Optimum Experimental Design in the Light of Pontryagin's Maximum Principle. SIAM Journal on Control and Optimization **51**(4), 3181–3207 (2013)
38. Sager, S., Claeys, M., Messine, F.: Efficient upper and lower bounds for global mixed-integer optimal control. Journal of Global Optimization **61**(4), 721–743 (2015). DOI 10.1007/s10898-014-0156-4
39. Sargent, R., Sullivan, G.: The development of an efficient optimal control package. In: J. Stoer (ed.) Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2. Springer, Heidelberg (1978)
40. Srinivasan, B., Bonvin, D.: Characterization of Optimal Temperature and Feed-Rate Policies for Discontinuous Two-Reaction Systems. Industrial & Engineering Chemistry Research **42**, 5607–5616 (2003)
41. Vasudevan, R., Gonzalez, H., Bajcsy, R., Sastry, S.S.: Consistent approximations for the optimal control of constrained switched systems—part 1: A conceptual algorithm. SIAM Journal on Control and Optimization **51**(6), 4463–4483 (2013)
42. Wächter, A., Biegler, L.: Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence. SIAM Journal on Computing **16**(1), 1–31 (2005)
43. Wächter, A., Biegler, L.: On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. Mathematical Programming **106**(1), 25–57 (2006)
44. Wächter, A., Biegler, L.T.: Line Search Filter Methods for Nonlinear Programming: Local Convergence. In: Technical Report. Technical Report RC23033 (W0312-090), TJ Watson Research Center, Yorktown (2003)
45. Wittmann, R.: blocksqp_2 reference release (2025). DOI 10.5281/zenodo.17457926. URL `https://doi.org/10.5281/zenodo.17457926`

## A Implementation

The original C++ source code was made available under https://github.com/djanka2/blockSQP. It must be linked with `qpOASES` [12], which requires linking with a sparse linear solver if sparse indefinite QPs are to be solved using its Schur-complement approach. `MUMPS` [1] was used for our experiments. An archive of the exact version used in this work is available under https://github.com/ReWittmann/blockSQP_reference and [22]. A build system with scripts for running numerical experiments and patches is available under https://github.com/ReWittmann/blockSQP_reference_build.

We have implemented and added the methods presented in this work into the original `blockSQP C++` package, among various minor changes. The example problems are implemented in `Python` using the `CasADi` [3] framework. `blockSQP 2` is called through a `pybind11` [20] based interface. The complete source code including a build system is available under https://github.com/ReWittmann/blockSQP_2, the version used for this study is archived under [45].

### A.1 `C++` package extensions

The new methods are implemented on top of existing `blockSQP` data structures and functions. For example, Algorithm 2 is implemented in the following class method. Steps and Lagrange gradient differences are named *delta*, *gamma* as opposed to $d, \gamma$. The rescaling factor $S_u$ is multiplied to the scaling factors of the free variables. Scaling factors of all variables are passed as the return argument `ret_SF`.

```
void SQPmethod::calc_free_variables_scaling(double *ret_SF){
    int nIt, pos, nfree = prob->nVar, ind_1, scfree,
         scdep, count_delta = 0, count_gamma = 0;
    double bardelta_u, bardelta_x, bargamma_u, bargamma_x,
         S_u, rgamma = 0., rdelta = 0.;

    if (prob->n_vblocks < 1) return;
    nfree = prob->nVar;
    for (int k = 0; k < prob->n_vblocks; k++){
        nfree -= prob->vblocks[k].size*int(prob->vblocks[k].dependent);
    }

    nIt = std::min(vars->n_scaleIt, 5);
    for (int j = 0; j < nIt; j++){
        bardelta_u = 0.; bardelta_x = 0.; bargamma_u = 0.; bargamma_x = 0.;
        scfree = 0; scdep = 0;
        pos = (vars->dg_pos - nIt + 1 + j + vars->dg_nsave)%vars->dg_nsave;
        ind_1 = 0;
        for (int k = 0; k < prob->n_vblocks; k++){
            for (int i = 0; i < prob->vblocks[k].size; i++){
                if (std::abs(vars->deltaMat(ind_1 + i, pos)) > 1e-8){
                    if (prob->vblocks[k].dependent){
                        bardelta_x += std::abs(vars->deltaMat(ind_1+i, pos));
                        bargamma_x += std::abs(vars->gammaMat(ind_1+i, pos));
                        scdep += 1;
                    }
                    else{
                        bardelta_u += std::abs(vars->deltaMat(ind_1+i, pos));
                        bargamma_u += std::abs(vars->gammaMat(ind_1+i, pos));
                        scfree += 1;
                    }
                }
            }
            ind_1 += prob->vblocks[k].size;
        }

        if (scdep > 0 && scfree > 0){
            bardelta_x /= scdep; bargamma_x /= scdep;
            bardelta_u /= scfree; bargamma_u /= scfree;
        }
        else{
            bardelta_u = 0.; bardelta_x = 1.0;
            bargamma_u = 0.; bargamma_x = 1.0;
        }
```

```
        if (bargamma_x > 5e-7 && bargamma_u > 5e-7){
            rgamma += std::log(bargamma_u/bargamma_x);
            count_gamma += 1;
            if (bardelta_x > 5e-7 && bardelta_u > 5e-7){
                rdelta += std::log(bardelta_u/bardelta_x);
                count_delta += 1;
            }
        }
    }
    rdelta = (count_delta > 0) ? std::exp(rdelta/count_delta) : 1.0;
    rgamma = (count_gamma > 0) ? std::exp(rgamma/count_gamma) : 1.0;

    S_u = -1.0;
    if (rgamma > 10.0){
        S_u = rgamma/10.0;
    }
    else if (rgamma < 1.0){
        if (rdelta > 1.0){
            if (rgamma < 0.1) S_u = 10.0*rgamma;
            else S_u = std::min(1.0, rdelta*rgamma);
        }
        else{
            S_u = rgamma;
        }
    }

    if (S_u > 0){
        vars->vfreeScale *= S_u;
        ind_1 = 0;
        for (int k = 0; k < prob->n_vblocks; k++){
            if (!prob->vblocks[k].dependent){
                for (int i = 0; i < prob->vblocks[k].size; i++){
                    ret_SF[ind_1 + i] *= S_u;
                }
            }
            ind_1 += prob->vblocks[k].size;
        }
    }
    return;
}
```

## A.2 Python interface and benchmark problems

The `Python` interface of `blockSQP` is mostly analogous to the `C++` interface.

```
import py_blockSQP
from blockSQP_pyProblem import blockSQP_pyProblem as Problemspec

opts = py_blockSQP.SQPoptions()
...
stats = py_blockSQP.SQPstats("./solver/output/path")
...
prob = Problemspec()
...
optimizer = py_blockSQP.SQPmethod(prob, opts, stats)
optimizer.init()
ret = optimizer.run(max_num_iterations)
optimizer.finish()
```

The benchmark problems are implemented in `CasADi` [3], employing a helper subclass and `NumPy` [18]. `Matplotlib` [19] is used for plotting.

```
import numpy as np
import casadi as cs
import matplotlib.pyplot as plt
...
class OCProblem:
```

```
        ...
...
class Lotka_Volterra_Fishing(OCProblem):
    default_params = {'c0':0.4, 'c1':0.2, 'x_init':[0.5,0.7],\
                      't0':0.0, 'tf':12.0}

    def build_problem(self):
        self.set_OCP_data(2,0,1,1,[0,0],[np.inf, np.inf],[],[],[0],[1])
        self.fix_time_horizon(self.model_params['t0'],self.model_params['tf'])
        self.fix_initial_value(self.model_params['x_init'])

        x = cs.MX.sym('x', 2)
        w = cs.MX.sym('w', 1)
        x0, x1 = cs.vertsplit(x)
        ode_rhs = cs.vertcat(
                x0 - x0*x1 - self.model_params['c0']*x0*w,
                -x1 + x0*x1 - self.model_params['c1']*x1*w
                )
        quad_expr = (x0 - 1)**2 + (x1 - 1)**2
        dt = cs.MX.sym('dt', 1)
        self.ODE = {
                'x': x,
                'p':cs.vertcat(dt, w),
                'ode': dt*ode_rhs,
                'quad': dt*quad_expr
                }
        self.multiple_shooting()
        self.set_objective(self.q_tf)
        self.build_NLP()

        self.start_point = np.zeros(self.nVar)
        for i in range(self.ntS+1):
            self.set_stage_state(self.start_point, i,\
                                 self.model_params['x_init'])
        for i in range(self.ntS):
            self.set_stage_control(self.start_point, i, [0])
    def plot(self, xi, dpi = None, title = None, it = None):
        ...
```

## B Benchmark problems

Below we describe all benchmark problems employed for our study. $S(u) \coloneqq S(u,...)$ refers to the solution function (1), which is obtained by integrating the states over the full time horizon and commonly used for initialization. Perturbed start points are obtained by applying a modification to each initial discretized control $u_k^{(init)}$ in turn.

### B.1 Batch reactor [34]

This example models the consecutive reaction of substance $A \to B \to C$. The goal is to maximize the amount of produced substance $B$. The control $u$ is the temperature.

$$
\begin{aligned}
\min_{x,u} \ & -x_2(t_F) \\
s.t. \ & \dot{x}_1 = -k_1 x_1^2 \\
& \dot{x}_2 = k_1 x_1^2 - k_2 x_2 \\
& x(t_0) = (1,0)^T \\
& k_1 = 4000 e^{(-2500/u)} \\
& k_2 = 620000 e^{(-5000/u)} \\
& 298 \leq u(t) \leq 398
\end{aligned}
$$

Parameters: $t_0 = 0, t_F = 1$
Start point: $u^{(init)} \equiv 298, x^{(init)} = S(u^{(init)})$
Perturbation: $u_k^{(init)} \leftarrow 300$

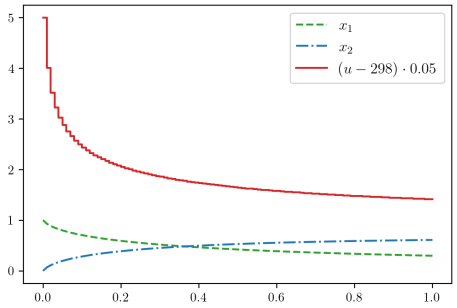



**Fig. 16** Reference solution of the Batch reactor problem

## B.2 Cart pendulum

This example models a pendulum attached to a cart, with the goal being to bring and maintain the pendulum in an upright position. The control is the cart's forward and backward acceleration. The well known dynamics are described e.g. in [33].

$$\min_{x,v,\theta,w,u} \int_0^{t_F} 10\cdot x(t)^2 + 50\cdot(\theta(t)-\pi)^2 + \lambda_u \cdot u(t)^2 dt$$

$$\begin{aligned} s.t.\ \ & \dot{x} = v \\ & \dot{\theta} = w \\ & \dot{v} = \frac{u + m\cdot g\cdot\cos(\theta)\cdot\sin(\theta) + m\cdot l\cdot w^2\cdot\sin(\theta)}{M + m\cdot(1-\cos(\theta)^2)} \\ & \dot{w} = \frac{-g\cdot\sin(\theta) - \dot{v}\cdot\cos(\theta))}{l} \\ & -u_{max} \le u(t) \le u_{max} \\ & -2 \le x(t) \le 2 \end{aligned}$$

Parameters: $t_F = 4.0, M = 1, m = 0.1, l = 1, g = 9.81, u_{max} = 30, \lambda_u = 0.5$
Start point: $u^{(init)} \equiv 0, (x^{(init)}, \theta^{(init)}, v^{(init)}, w^{(init)})^T = S(u^{(init)})$
Perturbation: $u_k^{(init)} \leftarrow -1.0$
Alternative parameter sets are $u_{max} = 15, \lambda_u = 0.05$; $u_{max} = 30, \lambda_u = 0.05$.

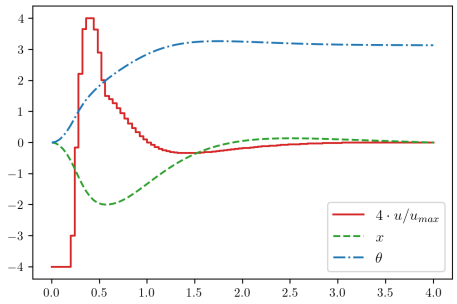



**Fig. 17** Reference solution of the Cart pendulum problem

## B.3 Catalyst mixing [10]

This example models a nonlinear reaction with the control determining the ratio of the two catalysts.

$$
\begin{aligned}
\min_{x,u} \quad & -1 + x_1(t_F) + x_2(t_F) \\
s.t. \quad & \dot{x}_1 = u \cdot (10x_2 - x_1) \\
& \dot{x}_2 = u \cdot (x_1 - 10x_2) - (1-u) \cdot x_2 \\
& 0 \le u(t) \le 1 \\
& x(t_0) = (1,0)^T
\end{aligned}
$$

Parameters: $t_0 = 0, t_1 = 1$
Start point: $u^{(init)} \equiv 0, x^{(init)} \equiv x(t_0)$
Perturbation: $u_k^{(init)} \leftarrow 0.1$

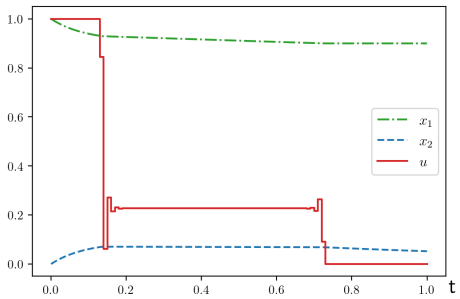



**Fig. 18** Reference solution of the Catalyst mixing problem

## B.4 Cushioned oscillation problem

This example models an oscillating mass attached to a spring where the control is a force that may be applied to the mass. The aim is to stop the oscillations in a minimal amount of time.

$$
\begin{aligned}
\min_{x,v,u,t_F} \quad & t_F \\
s.t. \quad & \dot{x} = v \\
& \dot{v} = \frac{1}{m}(u - c \cdot x) \\
& x(0) = x_0, \\
& v(0) = v_0, \\
& x(t_F) = 0, \\
& v(t_F) = 0, \\
& -u_{mm} \le u(t) \le u_{mm}
\end{aligned}
$$

Parameters:
$m = 5, c = 10, x_0 = 2, v_0 = 5, u_{mm} = 5$

Parameters: $m = 5, c = 10, x_0 = 2, v_0 = 5, u_{mm} = 5$
Start point: $u^{(init)} \equiv 0, x^{(init)} \equiv x_0, v^{(init)} \equiv v_0$
Perturbation: $u_k^{(init)} \leftarrow 0.1$

Due to the oscillating nature of the states, the optimizer could get stuck on the wrong cycle if an iteration

set $t_F$ below 8, leading to local infeasibility. Adding the bound $8 \geq t_F$ prevents this and does not affect the optimal solution where $t_F \approx 8.996$.

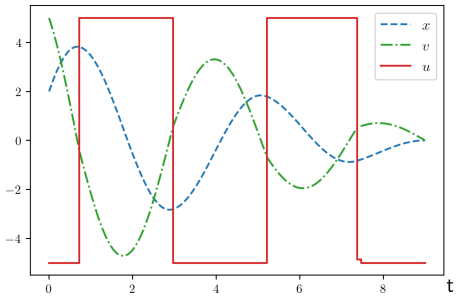



**Fig. 19** Reference solution of the Cushioned oscillation problem

## B.5 Egerstedt standard [11]

$$
\begin{aligned}
\min_{x,w}\ & x_3(t_F) \\
s.t.\ & \dot{x}_1 = -x_1 w_1 + (x_1 + x_2) w_2 + (x_1 - x_2) w_3 \\
& \dot{x}_2 = (x_1 + 2x_2) w_1 + (x_1 - 2x_2) w_2 + (x_1 + x_2) w_3 \\
& \dot{x}_3 = x_1^2 + x_2^2 \\
& 1 = \sum_{i=1}^{3} w_i(t) \\
& 0 \leq w(t) \leq 1 \\
& 0.4 \leq x_2(t) \\
& x(0) = (0.5, 0.5, 0)^T
\end{aligned}
$$

Parameters: $t_F = 1$
Start point: $w^{(init)} \equiv (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$, $x^{(init)} \equiv x(0)$
Perturbation: $w_k^{(init)} \leftarrow (0.5,\ 0.25,\ 0.25)^T$

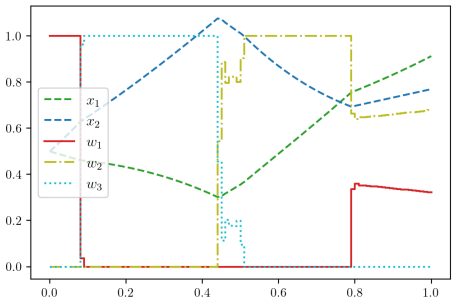



**Fig. 20** Reference solution of the Egerstedt standard problem

## B.6 Electric car [38]

In this example, the goal is to drive an electric car a fixed distance while minimizing the energy consumption.

$$
\begin{aligned}
\min_{x,u}\ & x_3(t_F) \\
s.t.\ & \dot{x}_0 = (V_{alim}u - R_m x_0 - K_m x_1)/L_m \\
& \dot{x}_1 = \frac{K_r^2}{Mr^2}(K_m x_0 - \frac{r}{K_r}(MgK_f + \frac{1}{2}\rho S C_x \frac{r^2}{K_r^2}x_1^2)) \\
& \dot{x}_2 = \frac{r}{K_r}x_1 \\
& \dot{x}_3 = V_{alim}ux_0 + R_{bat}x_0^2 \\
& x_2(t_f) = 100 \\
& -i_{max} \le x_0(t) \le i_{max} \\
& -1 \le u(t) \le 1 \\
& x(t_0) = (0,0,0,0)^T
\end{aligned}
$$

Parameters: $t_0 = 0$, $t_F = 1$, $K_r = 10$, $\rho = 1.293$, $C_x = 0.4$, $S = 2$, $r = 0.33$, $K_f = 0.03$, $K_m = 0.27$, $R_m = 0.03$, $L_m = 0.05$, $M = 250$, $g = 9.81$, $V_{alim} = 150$, $R_{bat} = 0.05$

Start point: $u^{(init)}(t) = 0.1 + 0.9 \cdot \frac{t-t_0}{t_F-t_0}$, $x^{(init)} = S(u^{(init)})$

Perturbation: $u_k^{(init)} \leftarrow 0.1$

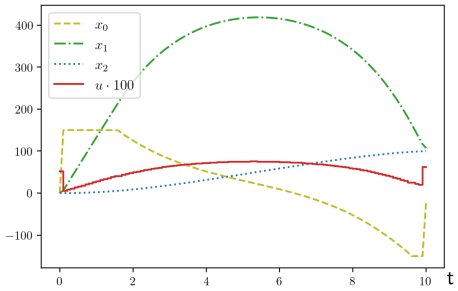



**Fig. 21** Reference solution of the Electric car problem

## B.7 Goddard's rocket

This is a variant of the well known problem pioneered by Goddard [17]. It models the ascent of a rocket through the atmosphere. The aim is to reach the target altitude while minimizing the fuel consumption.

$$
\begin{aligned}
\min_{m,r,v,u,t_F} \quad & -m(t_F) \\
s.t. \quad & \dot{r} = v \\
& \dot{v} = -\frac{1}{r^2} + \frac{1}{m}(T_{max}u - D(r,v)) \\
& \dot{m} = -b \cdot u \\
& r(t_F) = r_T \\
& D(r(t), v(t)) \leq C \\
& 0 \leq u(t) \leq 1 \\
& r(0) = r_0 \\
& v(0) = v_0 \\
& m(0) = m_0 \\
& D(r,v) := Av^2\rho(r) && \text{(drag)} \\
& \rho(r) := \exp(-k \cdot (r - r_0)) && \text{(atmospheric density)}
\end{aligned}
$$

Parameters: $r_0 = 1$, $v_0 = 0$, $m_0 = 1$, $r_T = 1.01$, $b = 7$,
$T_{max} = 3.5$, $A = 310$, $k = 500$, $C = 0.6$

Start point: $t_F^{(init)} = \frac{0.4}{b} \cdot 2.5$, $u^{(init)}(t) = \begin{cases} 1 & t \leq 0.4 \cdot t_F^{(init)} \\ 0 & t > 0.4 \cdot T^{(init)} \end{cases}$,
$r^{(init)}, v^{(init)}, m^{(init)} = S(u^{(init)})$

Perturbation: $u_k^{(init)} \leftarrow \begin{cases} 0.9 & u_k^{(init)} = 1 \\ 0.1 & u_k^{(init)} = 0 \end{cases}$

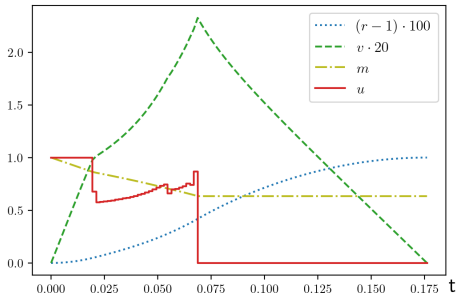



**Fig. 22** Reference solution of Goddard's rocket problem

## B.8 Hang glider [10]

This example models a hang glider in a thermal updraft with the goal of maximizing the final horizontal position.

$$
\begin{aligned}
\min_{x,v_x,y,v_y,c_L,t_F} \quad & -x(t_F) \\
s.t.\ \ & \dot{x} = v_x \\
& \dot{y} = v_y \\
& \dot{v_x} = \frac{1}{m}(-L\cdot\frac{w}{v} - D\cdot\frac{v_x}{v}) \\
& \dot{v_y} = \frac{1}{m}(L\cdot\frac{v_x}{v} - D\cdot\frac{w}{v}) - g \\
& 0 \le c_L(t) \le c_{max} \\
& v_x(t_F) = v_x(0) = 13.23 \\
& y(t_F) = 900 \\
& v_y(t_F) = v_y(0) = -1.288 \\
& x(0) = 0 \\
& y(0) = 1000 \\
& \\
& L := L(x,v_x,v_y,c_L) = \frac{1}{2}u\cdot\rho\cdot S\cdot v^2 \\
& D := D(x,v_x,v_y,c_L) = \frac{1}{2}(c_0 + c_1\cdot c_L^2)\cdot\rho\cdot S\cdot v \\
& v := v(x,v_x,v_y) = \sqrt{v_x^2 + (v_y - U)^2} \\
& U := U(x) = u_C(1-r)\exp(-r) \qquad \text{(Position dependent updraft)} \\
& r := r(x) = (\frac{x}{r_C} - 2.5)^2
\end{aligned}
$$

Parameters: $c_0 = 0.034$, $c_1 = 0.069662$, $S = 14$, $\rho = 1.13$,
$u_C = 2.5$, $r_C = 100$, $c_{max} = 1.4$, $m = 100$, $g = 9.81$

Start point: $t_F^{(init)} = 100$, $c_L^{(init)} \equiv c_{max}$, $(x^{(init)}, y^{(init)}, v_x^{(init)}, v_y^{(init)})^T = S(c_L^{(init)}, t_F^{(init)})$

Perturbation: $c_{L,k}^{(init)} \leftarrow c_{max} - 0.1$

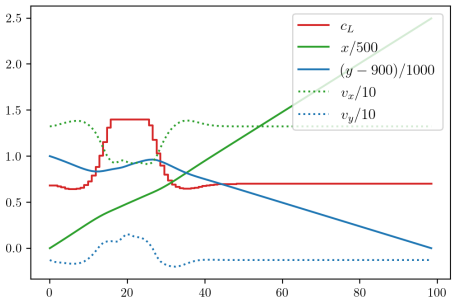



**Fig. 23** Reference solution of the Hang glider problem

## B.9 Hanging chain [10]

This example models a chain of length $L$ suspended between two points $z_0$, $z_F$ at heights $a$, $b$. The trajectory of the chain minimizes the potential energy. This can be modeled as the following optimal control problem.

$$
\begin{aligned}
\min_{x,u}\ & x_2(z_F) \\
s.t.\ & \frac{\mathrm{d}}{\mathrm{d}z}x_1 = u \\
& \frac{\mathrm{d}}{\mathrm{d}z}x_2 = x_1(1+u^2)^{1/2} \\
& \frac{\mathrm{d}}{\mathrm{d}z}x_3 = (1+u^2)^{1/2} \\
& x(z_0) = (a,0,0)^T \\
& x_1(z_F) = b \\
& x_3(z_F) = L \\
& 0 \le x(z) \le 10 \\
& -10 \le u(z) \le 20
\end{aligned}
$$

Parameters: $z_0 = 0$, $z_F = 1$, $a = 1$, $b = 3$, $L = 4$

Start point [10]: $x_1^{(init)}(z) = (2|b-a|)\cdot z\cdot(z-2z_m),\ z_m \coloneqq \begin{cases} 0.25 & b > a \\ 0.75 & \text{else} \end{cases}$

$u^{(init)} = \dot{x}_1^{(init)}, x_2^{(init)} = x_1^{(init)} u^{(init)}, x_3^{(init)} = u^{(init)}$

Perturbation: $u_k^{(init)} \leftarrow u_k^{(init)} + 0.1$

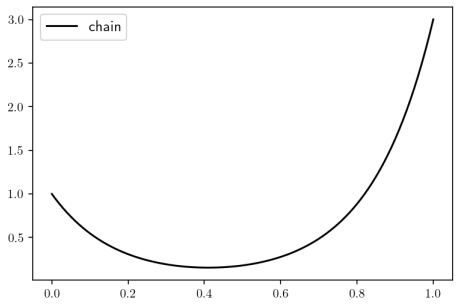



**Fig. 24** Reference solution of the Hanging chain problem

## B.10 Lotka Volterra optimum experimental design (Lotka OED) [35]

This is an optimum experimental design problem for the well known Lotka-Volterra predator-prey model. The aim is to find a control $u$ and measurement functions $w_1, w_2$ for measuring $p_2, p_4$ such that the confidence

region is small.

$$
\begin{aligned}
\min_{x,G,F,u,w} \quad & \operatorname{trace}(F^{-1}(t_F)) \\
s.t. \quad & \dot{x_1} = p_1\, x_1 - p_2 x_1 x_2 - p_5 u x_1 \\
& \dot{x_2} = -p_3\, x_2 + p_4 x_1 x_2 - p_6 u x_2 \\
& \dot{G_{11}} = (p_1 - p_2 x_2 - p_5 u) \cdot\ G_{11} + (-p_2 x_1) \cdot\ G_{21} + (-x_1 x_2) \cdot \\
& \dot{G_{12}} = (p_1 - p_2 x_2 - p_5 u) \cdot\ G_{12} + (-p_2 x_1) \cdot\ G_{22} \\
& \dot{G_{21}} = (p_4 x_2) \cdot\ G_{11} + (-p_3 + p_4 x_1) \cdot\ G_{21} \\
& \dot{G_{22}} = (p_4 x_2) \cdot\ G_{12} + (-p_3 + p_4 x_1) \cdot\ G_{22} + (x_1 x_2) \cdot \\
& \dot{F_{11}} = w_1 G_{11}^2 + w_2 G_{21}^2 \\
& \dot{F_{12}} = \dot{F_{21}} = w_1 G_{11} G_{12} + w_2 G_{21} G_{22} \\
& \dot{F_{22}} = w_1 G_{12}^2 + w_2 G_{22}^2 \\
& \dot{z_1} = w_1 \\
& \dot{z_1} = w_2 \\
& 0 \le u(t) \le 1 \\
& 0 \le w(t) \le 1 \\
& 0 \le z(t_F) \le M \\
& x(0) = (0.5, 0.7) \\
& G(0) = F(0) = 0 \\
& z_1(0) = z_2(0) = 0
\end{aligned}
$$

Parameters: $t_F = 12$, $p_1 = p_2 = p_3 = p_4 = 1$, $p_5 = 0.4$, $p_6 = 0.2$, $M = 4$
Start point: $u^{(init)} \equiv 0$, $w_1 \equiv \frac{1}{3}$, $w_2 \equiv \frac{1}{3}$, $(x^{(init)}, G^{(init)}, F^{(init)})^T = S(u^{(init)}, w^{(init)})$
Perturbation: $u_k^{(init)} \leftarrow 0.1$

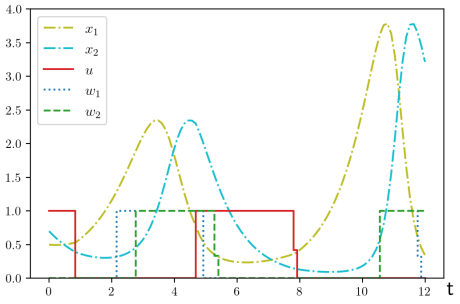



**Fig. 25** Reference solution of the Lotka OED problem

## B.11 Lotka Volterra fishing [35]

This example is an optimal control problem for the well known Lotka-Volterra predator-prey model, which aim to bring the dynamics to a steady state.

$$
\begin{aligned}
\min_{x,u} \quad & \int_0^{t_F} (x_1(t)-1)^2 + (x_2(t)-1)^2 dt \\
s.t. \quad & \dot{x_1} = p_1\, x_1 - p_2 x_1 x_2 - p_5 u x_1 \\
& \dot{x_2} = -p_3\, x_2 + p_4 x_1 x_2 - p_6 u x_2 \\
& 0 \le u(t) \le 1 \\
& x(0) = (0.5, 0.7)^T
\end{aligned}
$$

Parameters: $t_F = 12$, $p_1 = p_2 = p_3 = p_4 = 1$, $p_5 = 0.4$, $p_6 = 0.2$
Start point: $u^{(init)} \equiv 0$, $x^{(init)} \equiv x(0)$
Perturbation: $u_k^{(init)} \leftarrow 0.1$

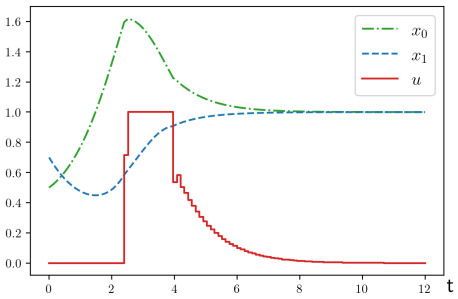



**Fig. 26** Reference solution of the Lotka Volterra fishing problem

## B.12 Particle steering [10]

This example models a particle steered by a constant force, with the control being the thrust angle. The aim is to achieve a given terminal altitude and velocity in minimal time.

$$
\begin{aligned}
\min_{x,u} \quad & t_F \\
s.t. \quad & \ddot{x}_1 = a \cdot \cos(u) \\
& \ddot{x}_2 = a \cdot \sin(u) \\
& x_1(0), \dot{x}_1(0), x_2(0), \dot{x}_2(0) = 0 \\
& x_2(t_F) = 5 \\
& \dot{x}_1(t_F), \dot{x}_2(t_F) = 45, 0 \\
& -\frac{\pi}{2} \le u(t) \le \frac{\pi}{2}
\end{aligned}
$$

Parameters: $a = 100$
Start point: $t_F^{(init)} = 1.0$, $u^{(init)} \equiv 0$, $x^{(init)} \equiv x(0)$, $\dot{x}^{(init)} \equiv \dot{x}(0)$
Perturbation: $u_k^{(init)} \leftarrow 0.1$

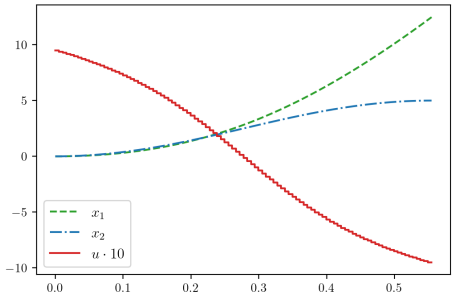



**Fig. 27** Reference solution of the Particle steering problem

## B.13 Quadrotor helicopter [16]

This example models a quadrotor helicopter in two dimensions. The states are vertical position, horizontal position, roll angle and their derivatives.

$$\min_{x,u,w} \; 5(x_1(t_f)-6)^2 + 5(x_3(t_f)-1)^2 + (\frac{1}{2}\sin(x_5(t_f)))^2 + \int_{t_0}^{t_f} 5u(\tau)^2 \, d\tau$$

$$\begin{aligned}
s.t.\;\; \dot{x}_1 &= x_2 && \text{(horizontal position)}\\
\dot{x}_2 &= g\sin(x_5) + w_1 u \frac{\sin(x_5)}{M}\\
\dot{x}_3 &= x_4 && \text{(vertical position)}\\
\dot{x}_4 &= g\cos(x_5) - g + w_1 u \frac{\cos(x_5)}{M}\\
\dot{x}_5 &= x_6 && \text{(roll angle)}\\
\dot{x}_6 &= -w_2 L\frac{u}{I} + w_3 L\frac{u}{I}\\
0 &\le w_1(t), w_2(t), w_3(t) \le 1\\
w_1(t) + w_2(t) + w_3(t) &= 1\\
0 &\le u(t) \le 0.001\\
0 &\le x_3(t)\\
x(0) &= (0,0,1,0,0,0)^T
\end{aligned}$$

Parameters: $g = 9.8, M = 1.3, L = 0.305, I = 0.0605$
Start point: $w^{(init)} \equiv (0.5, 0., 0.5), u^{(init)} \equiv 0.001, x^{(init)} = S(w^{(init)}, u^{(init)})$
Perturbation: $w_k^{(init)} \leftarrow (0.45, 0.1, 0.45)^T$

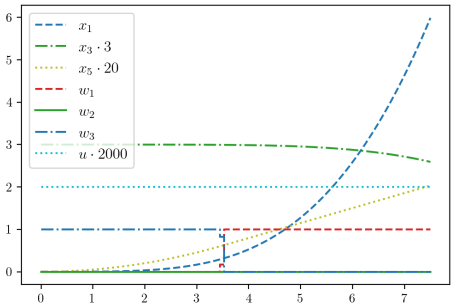



**Fig. 28** Reference solution of the Quadrotor helicopter problem

## B.14 Three tank multimode

This example is a variant of the two tank problem discussed in [41]. It models a system of three tanks, with the inflow into the first tank and the flow from the first to the third tank being controllable. The aim is to keep consistent fluid levels in tank 2 and tank 3.

$$
\begin{aligned}
\min_{x,w} \quad & \int_0^T k_1(x_2(t)-k_2)^2 + k_3(x_3(t)-k_4)^2 dt \\
s.t. \quad & \dot{x}_1 = -\sqrt{x_1} + c_1 w_1 + c_2 w_2 - w_3\sqrt{c_3 x_1} \\
& \dot{x}_2 = \sqrt{x_1} - \sqrt{x_2} \\
& \dot{x}_3 = \sqrt{x_2} - \sqrt{x_3} + w_3\sqrt{c_3 x_1} \\
& x(0) = (2,2,2)^T \\
& w_1(t) + w_2(t) + w_3(t) = 1 \\
& 0 \le x_1(t), x_2(t), x_3(t) \\
& 0 \le w_1(t), w_2(t), w_3(t) \le 1
\end{aligned}
$$

Parameters: $T = 12$, $c_1 = 1$, $c_2 = 2$, $c_3 = 0.8$, $k_1 = 2$, $k_2 = 3$, $k_3 = 1$, $k_4 = 3$
Start point: $w^{(init)} \equiv (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $x^{(init)} = S(w^{(init)}, u^{(init)})$
Perturbation: $w_k^{(init)} \leftarrow (0.45, 0.1, 0.45)^T$

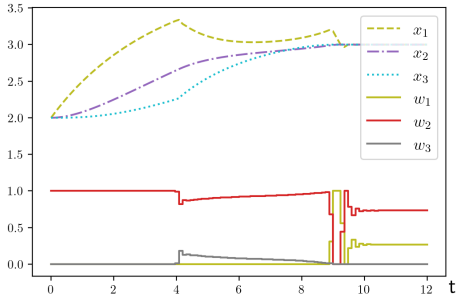



**Fig. 29** Reference solution of the Three tank multimode problem

## B.15 Time optimal car [26]

This example models a car with the goal of driving it a distance of 300 meters in minimum time.

$$
\begin{aligned}
\min_{z,u,t_f} \quad & t_f \\
s.t. \quad & \dot{z}_1 = z_2 \\
& \dot{z}_2 = u \\
& 0 \le z_1(t) \le 33 \\
& 0 \le z_2(t) \le 330 \\
& -2 \le u(t) \le 1 \\
& z(t_f) = (300,0)^T \\
& z(0) = (0,0)^T
\end{aligned}
$$

Parameters:
Start point: $t_f^{(init)} = 10,\ u^{(init)} \equiv 0,\ z^{(init)} \equiv z(0)$
Perturbation: $u_k^{(init)} \leftarrow 0.1$

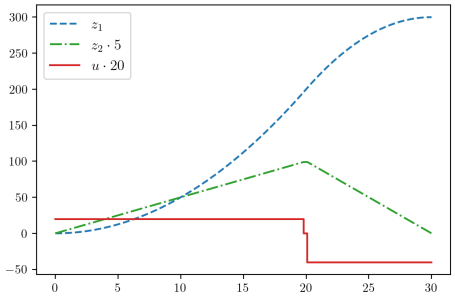



**Fig. 30** Reference solution of the Time optimal car problem

## B.16 Tubular reactor problem

This example was provided as a tutorial problem for the `GEKKO` optimization suite [4].

$$\begin{aligned} \min_{x,w} \ & -\int_0^{t_F} w(t) \cdot x_1(t) dt \\ s.t. \ & \dot{x} = -(w + \frac{1}{2} w^2) \cdot x \\ & 0 \le w(t) \le 5.0 \\ & x(0) = 1 \end{aligned}$$

Parameters: $t_F = 1$
Start point: $w^{(init)} \equiv 5,\ x^{(init)} \equiv 0$
Perturbation: $w_k^{(init)} \leftarrow 4.9$

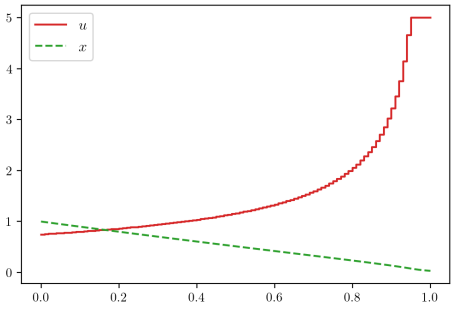



**Fig. 31** Reference solution of the Tubular reactor problem

# C Complete numerical results

Below are the complete numerical results of the experiments described in Sect. 4, 5 and 6.

## C.1 Comparison of convexification strategies

Table 1 contains the detailed results of the numerical experiments for the convexification strategies of Sect. 4.3, which were displayed in Fig. 6. They include empirical standard deviations as a measure for consistency.

| | $\mu$, $\sigma$ | SR1-BFGS | conv. str. 1 | conv. str. 2 |
|---|---|---|---|---|
| Batch reactor | N | 28.75, 5.50 | 20.12, 0.84 | 19.6, 1.09 |
| | t | 0.38s, 0.048s | 0.27s, 0.0088s | 0.27s, 0.011s |
| Cart pendulum | N | 73.10, 8.94 | 56.42, 3.31 | 54.55, 3.13 |
| | t | 1.43s, 0.14s | 1.39s, 0.075s | 1.36s, 0.078s |
| Catalyst mixing | N | 144.65, 50.64† | 23.45, 3.71 | 25.12, 3.66 |
| | t | 1.54s, 0.93s† | 0.24s, 0.036s | 0.26s, 0.041s |
| Cushioned oscill. | N | 27.32, 2.48 | 26.60, 3.59 | 26.75, 3.23 |
| | t | 0.43s, 0.041s | 1.22s, 0.27s | 1.26s, 0.27s |
| Egerstedt standard | N | 23.67, 3.01 | 21.45, 2.10 | 21.05, 1.16 |
| | t | 0.52s, 0.041s | 0.67s, 0.092s | 0.59s, 0.056s |
| Electric car | N | 13.55, 0.86 | 12.80, 1.16 | 13.55, 0.86 |
| | t | 0.22s, 0.014s | 0.26s, 0.034s | 0.25s, 0.017s |
| Goddard's rocket | N | 30.75, 10.49 | 17.82, 1.62 | 17.57, 1.89 |
| | t | 0.62s, 0.15s | 0.50s, 0.049s | 0.49s, 0.056s |
| Hang glider | N | 54.87, 13.42 | 49.50, 6.05 | 58.40, 22.96 |
| | t | 1.85s, 1.01s | 2.03s, 0.62s | 2.85s, 3.98s |
| Hanging chain | N | 150.85, 67.29† | 24.52, 4.99 | 22.02, 3.26 |
| | t | 1.34s, 0.75s† | 0.18s, 0.042s | 0.16s, 0.028s |
| Lotka OED | N | 95.62, 29.35 | 72.20, 8.02 | 39.40, 5.35 |
| | t | 6.02s, 1.24s | 11.20s, 2.01s | 8.32s, 1.65s |
| Lotka Volterra fishing | N | 15.45, 1.07 | 15.90, 0.99 | 15.77, 0.93 |
| | t | 0.35s, 0.016s | 0.39s, 0.025s | 0.40s, 0.023s |
| Particle steering | N | 25.72, 2.50 | 36.25, 9.49 | 27.27, 2.33 |
| | t | 0.94s, 0.21s | 1.33s, 0.34s | 1.04s, 0.16s |
| Quadrotor helicopter | N | 8.27, 0.54 | 8.40, 1.04 | 8.40, 1.04 |
| | t | 1.46s, 0.20s | 1.78s, 0.21s | 1.74s, 0.20s |
| Three tank multimode | N | 16.38, 1.32 | 16.77, 1.11 | 16.93, 1.13 |
| | t | 0.60s, 0.03s | 0.60s, 0.03s | 0.63s, 0.04s |
| Time optimal car | N | 11.62, 1.51 | 11.95, 1.64 | 11.77, 1.54 |
| | t | 1.96s, 0.52s | 2.37s, 0.83s | 2.33s, 0.82s |
| Tubular reactor | N | 27.82, 4.01 | 24.25, 3.44 | 24.15, 3.51 |
| | t | 0.33s, 0.38s | 0.22s, 0.030s | 0.21s, 0.024s |

**Table 1** Total iterations N solution times t with and without convexification strategies for sequential solution of QPs
† unsuccessful termination for most start points

## C.2 Sequential-parallel comparison

Table 2 below shows the exact results of the comparison between sequential and parallel solution of QPs that was performed in Section 4.3 and shown in Fig. 7.

| | $\mu$, $\sigma$ | conv. str. 2, sequential | conv. str. 2, parallel |
|---|---|---|---|
| Batch reactor | N | 19.6, 1.09 | 19.9, 1.04 |
| | t | 0.28s, 0.014s | 0.34s, 0.034s |
| Cart pendulum | N | 54.55, 3.08 | 57.22, 3.65 |
| | t | 1.36s, 0.075s | 1.16s, 0.063s |
| Catalyst mixing | N | 25.12, 3.66 | 24.95, 4.12 |
| | t | 0.26s, 0.041s | 0.25s, 0.038s |
| Cushioned oscill. | N | 26.75, 3.38 | 28.77, 2.70 |
| | t | 1.26s, 0.27s | 0.70s, 0.078s |
| Egerstedt standard | N | 21.05, 1.16 | 22.3, 2.09 |
| | t | 0.59s, 0.055s | 0.55s, 0.039s |
| Electric car | N | 13.55, 0.86 | 13.55, 0.86 |
| | t | 0.26s, 0.017s | 0.23s, 0.015s |
| Goddard's rocket | N | 17.57, 1.89 | 17.95, 2.06 |
| | t | 0.49s, 0.056s | 0.46s, 0.035s |
| Hang glider | N | 58.4, 22.96 | 59.97, 27.30 |
| | t | 2.84s, 3.97s | 2.39s, 3.11s |
| Hanging chain | N | 22.02, 3.26 | 19.2, 2.92 |
| | t | 0.16s, 0.028s | 0.16s, 0.041s |
| Lotka OED | N | 39.4, 5.35 | 40.17, 4.93 |
| | t | 8.25s, 1.63s | 4.47s, 0.49s |
| Lotka Volterra fishing | N | 15.77, 0.93 | 15.92, 0.98 |
| | t | 0.39s, 0.022s | 0.37s, 0.018s |
| Particle steering | N | 27.27, 2.33 | 27.77, 3.25 |
| | t | 1.07s, 0.16s | 0.94s, 0.18s |
| Quadrotor helicopter | N | 8.4, 1.04 | 8.4, 1.04 |
| | t | 1.75s, 0.21s | 1.51s, 0.25s |
| Three tank (multimode) | N | 16.93, 1.13 | 17.23, 1.25 |
| | t | 0.63s, 0.04s | 0.61s, 0.03s |
| Time optimal car | N | 11.77, 1.54 | 11.77, 1.54 |
| | t | 2.31s, 0.81s | 1.92s, 0.38s |
| Tubular reactor | N | 24.1, 3.55 | 24.22, 2.55 |
| | t | 0.21s, 0.024s | 0.18s, 0.022s |

**Table 2** Total iterations N solution times t for sequential and parallel solution of QPs

## C.3 Scaling comparison

Table 3 below shows the exact results of benchmark performed for the scaling heuristic proposed in Section 5.4 and previously displayed in Fig. 12.

| | $\mu$, $\sigma$ | no scaling | automatic scaling |
|---|---|---|---|
| Batch reactor | N | 19.8, 1.02 | 19.82, 1.94 |
| | t | 0.34s, 0.024s | 0.31s, 0.023s |
| Cart pendulum | N | 57.05, 3.17 | 55.12, 3.37 |
| | t | 1.15s, 0.059s | 1.13s, 0.063s |
| Catalyst mixing | N | 24.8, 3.78 | 25.82, 4.62 |
| | t | 0.25s, 0.047s | 0.26s, 0.052s |
| Cushioned oscill. | N | 28.37, 3.07 | 36.25, 8.61 |
| | t | 0.70s, 0.091s | 0.97s, 0.26s |
| Egerstedt standard | N | 22.22, 1.50 | 23.92, 2.38 |
| | t | 0.55s, 0.029s | 0.59s, 0.042s |
| Electric car | N | 13.65, 0.96 | 11.67, 0.64 |
| | t | 0.23s, 0.028s | 0.20s, 0.012s |
| Goddard's rocket | N | 18.07, 2.10 | 17.90, 1.94 |
| | t | 0.46s, 0.037s | 0.46s, 0.032s |
| Hang glider | N | 60.97, 27.17 | 45.47, 12.03 |
| | t | 2.42s, 3.09s | 1.49s, 0.99s |
| Hanging chain | N | 19.60, 3.09 | 20.20, 4.50 |
| | t | 0.17s, 0.046s | 0.17s, 0.046s |
| Lotka OED | N | 40.47, 4.93 | 38.00, 4.14 |
| | t | 4.50s, 0.52s | 4.44s, 0.49s |
| Lotka Volterra fishing | N | 15.87, 0.97 | 9.42, 0.54 |
| | t | 0.37s, 0.015s | 0.28s, 0.0050s |
| Particle steering | N | 27.77, 3.25 | 28.02, 2.91 |
| | t | 0.93s, 0.18s | 0.48s, 0.10s |
| Quadrotor helicopter | N | 8.40, 1.04 | 5.95, 0.99 |
| | t | 1.52s, 0.25s | 1.66s, 0.097s |
| Three tank multimode | N | 17.30, 1.17 | 12.68, 1.29 |
| | t | 0.61s, 0.03s | 0.54s, 0.04s |
| Time optimal car | N | 11.77, 1.54 | 11.97, 1.69 |
| | t | 1.92s, 0.38s | 2.01s, 0.38s |
| Tubular reactor | N | 24.10, 3.55 | 24.22, 2.55 |
| | t | 0.21s, 0.024s | 0.18s, 0.022s |

**Table 3** Iteration counts N and solution times with and without automatic scaling

## C.4 `blockSQP` - `ipopt` - `orig. blockSQP` comparison

Table 4 below shows the averaged total iterations and solution time of `ipopt`, `blockSQP` and the `original blockSQP`. `ipopt` and `blockSQP` were tested for both quasi-Newton and the exact Hessian, `original blockSQP` for SR1-BFGS and the original convexification strategy. We highlighted the runtime comparison between exact Hessian `ipopt` and quasi-Newton `blockSQP`.

| | | `ipopt` | | `blockSQP` | | `orig. blockSQP` | |
|---|---|---|---|---|---|---|---|
| | $\mu$ | QN | e.H. | QN | e.H. | SR1-BFGS | conv. str. |
| Batch reactor | N | 16.92 | 9.15 | 19.82 | 12.0 | 63.12 | 37.05 |
| | t | 0.20s | **0.10s** | 0.31s | 0.38s | 3.37s | 0.61s |
| Cart pendulum | N | 89.47 | 94.07 | 55.42 | 41.25 | 86.10 | 58.02 |
| | t | 2.00s | 2.09s | **1.13s** | 1.89s | 3.46s | 2.82s |
| Catalyst mixing | N | 23.5 | 14.0 | 25.52 | 7.22 | 78.10† | 77.97† |
| | t | 0.27s | **0.13s** | 0.26s | 0.18s | 6.91s† | 8.93s† |
| Cushioned oscill. | N | 665.47 | 54.65 | 36.02 | 21.57 | 13.12 | 12.70 |
| | t | 4.80s | **0.42s** | 0.98s | 0.84s | 0.27s | 0.40s |
| Egerstedt standard | N | 50.62 | 29.0 | 23.7 | 16.55 | 24.95 | 22.02 |
| | t | 0.86s | **0.51s** | 0.58s | 0.71s | 0.51s | 0.57s |
| Electric car | N | 17.87 | 15.0 | 11.67 | 5.0 | 13.02 | 19.25 |
| | t | 0.31s | 0.22s | **0.21s** | 0.17s | 0.24s | 1.06s |
| Goddard's rocket | N | 77.35 | 124.25 | 18.0 | 6.02 | 34.20 | 20.50 |
| | t | 1.09s | 2.73s | **0.46s** | 0.35s | 1.41s | 0.74s |
| Hang glider | N | 1000.0† | 66.82 | 45.47 | 33.57 | 80.75 | 54.42 |
| | t | 23.42s† | 1.91s | **1.50s** | 6.74s | 5.45s | 4.21s |
| Hanging chain | N | 71.27 | 26.05 | 20.17 | 9.02 | 42.27 | 30.60 |
| | t | 0.74s | 0.26s | **0.17s** | 0.16s | 2.03s | 0.55s |
| Lotka OED | N | 45.95 | 30.0 | 37.7 | 14.7 | 131.32† | 71.35 |
| | t | 1.62s | **1.81s** | 4.40s | 3.55s | 39.01s† | 17.60s |
| Lotka Volterra fishing | N | 105.22 | 21.55 | 9.42 | 5.0 | 22.55 | 19.57 |
| | t | 1.49s | **0.25s** | 0.28s | 0.28s | 0.54s | 0.71s |
| Particle steering | N | 94.05 | 16.9 | 27.97 | 14.4 | 24.90 | 25.15 |
| | t | 1.10s | **0.15s** | 0.47s | 0.48s | 1.19s | 1.37s |
| Quadrotor helicopter | N | 62.07 | 59.0 | 5.95 | 5.47 | 14.0 | 12.72 |
| | t | 1.47s | 1.83s | **1.66s** | 1.82s | 1.75s | 1.84s |
| Three tank | N | 316.07 | 15.0 | 12.57 | 7.0 | 22.75 | 19.2 |
| | t | 7.16s | **0.31s** | 0.54s | 0.55s | 0.85s | 1.11s |
| Time optimal car | N | 50.15 | 86.52 | 11.97 | 9.42 | 40.47 | 40.47 |
| | t | 0.45s | **0.57s** | 2.01s | 1.92s | 3.83s | 6.24s |
| Tubular reactor | N | 54.87 | 16.05 | 24.57 | 19.17 | 56.05 | 55.92 |
| | t | 0.49s | **0.13s** | 0.20s | 0.39s | 1.67s | 2.33s |

**Table 4** Iteration counts N and solution times for `ipopt`, `blockSQP` and the original version of `blockSQP`, using an explicit Runge-Kutta-4 ODE integrator. `blockSQP` and `ipopt` are run both with quasi-Newton and the exact Hessian, `original blockSQP` for both SR1-BFGS and the original convexification strategy described in Section 4.2.
† unsuccessful termination for most start points

Table 5 below shows the performance of `blockSQP` and `ipopt` for a fixed step implicit integrator.

| | | `ipopt` | | `blockSQP` | |
|---|---|---|---|---|---|
| | $\mu$ | QN | e.H. | QN | e.H. |
| Batch reactor | N | 16.92 | 9.15 | 19.87 | 12.0 |
| | t | 0.74s | **0.58s** | 1.07s | 1.05s |
| Cart pendulum | N | 87.62 | 94.02 | 54.87 | 40.6 |
| | t | 8.84s | 13.86s | **5.18s** | 6.95s |
| Catalyst mixing | N | 23.57 | 14.0 | 25.02 | 7.22 |
| | t | 0.76s | **0.65s** | 0.85s | 0.45s |
| Cushioned oscill. | N | 718.22 | 54.65 | 36.4 | 21.15 |
| | t | 12.63s | 1.75s | **1.35s** | 1.36s |
| Egerstedt standard | N | 49.87 | 29.0 | 23.52 | 16.42 |
| | t | 2.41s | 2.32s | **1.30s** | 1.66s |
| Electric car | N | 17.85 | 15.0 | 11.82 | 5.0 |
| | t | 1.14s | 1.33s | **0.79s** | 0.49s |
| Goddard's rocket | N | 83.32 | 106.9 | 17.9 | 6.02 |
| | t | 4.48s | 12.68s | **1.40s** | 0.86s |
| Hang glider | N | - | 68.10 | 51.83 | 25.80 |
| | t | - | 13.03s | **9.74s** | 6.36s |
| Hanging chain | N | 71.27 | 26.05 | 19.77 | 9.02 |
| | t | 2.23s | 1.28s | **0.60s** | 0.45s |
| Lotka OED | N | 46.35 | 30.0 | 38.55 | 14.72 |
| | t | 9.92s | 15.80s | **11.36s** | 9.87s |
| Lotka Volterra fishing | N | 105.4 | 21.55 | 9.47 | 5.0 |
| | t | 6.73s | 1.77s | **0.71s** | 0.60s |
| Particle steering | N | 94.05 | 16.9 | 28.72 | 14.42 |
| | t | 2.23s | **0.64s** | 0.90s | 0.89s |
| Quadrotor helicopter | N | 60.07 | 59.0 | 6.0 | 5.5 |
| | t | 4.56s | 8.61s | **2.22s** | 2.50s |
| Three tank | N | 323.20 | 15.0 | 12.68 | 7.0 |
| | t | 29.20s | 1.82s | **1.24s** | 1.16s |
| Time optimal car | N | 50.25 | 86.52 | 12.4 | 9.45 |
| | t | 0.96s | 2.53s | **2.25s** | 2.20s |
| Tubular reactor | N | 40.62 | 15.07 | 26.1 | 18.97 |
| | t | 1.14s | **0.68s** | 0.75s | 1.05s |

**Table 5** Iteration counts $N$ and solution times for `ipopt` and `blockSQP`, using a fixed step implicit ODE integrator. The solvers are tested for both quasi-Newton and the exact Hessian

## D P2Chem Problem

This is an OCP based on the model presented in [14]. It describes a system reactors connected by channels, along with sources, sinks, mixers and splitters.

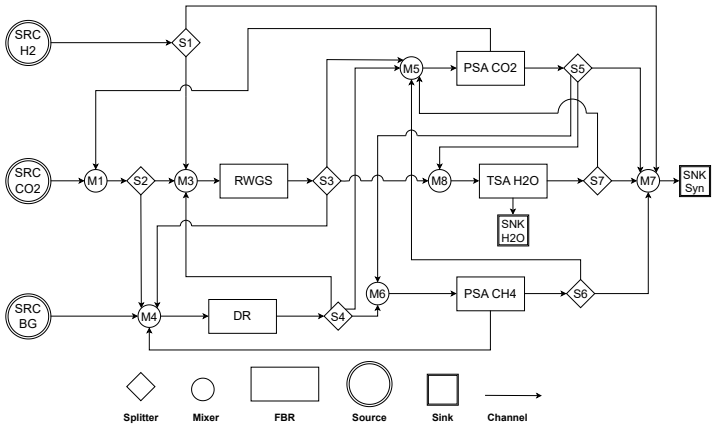



**Fig. 32** The reactor network consisting of different units. The fixed bed reactors (FBRs) are modeled via partial differential equation semidiscretized into differential algebraic equations (DAEs)

Each unit has its own variables, equations and constraints. Two connected units A and B are coupled linearly via the OCP constraints

$$\text{outflow(A)} = \text{inflow(B)}.$$

The 5 reactors are modeled by partial differential equations and semidiscretized into differential algebraic equations of the form

$$\dot{x}_{k,\alpha} = -v_k \frac{x_{k,\alpha} - x_{k-1,\alpha}}{\Delta_z} + D \frac{x_{k-1,\alpha} - 2x_{k,\alpha} + x_{k+1,\alpha}}{(\Delta_z)^2} + C_\sigma \left( \sigma_\alpha(x_k) - x_{k,\alpha} \sum_{\beta \in S} \sigma_\beta(x_k) \right)$$

$$v_k = v_{k-1} + \Delta_z C_\sigma \sum_{\alpha \in S} \sigma_\alpha(x_{k-1}), \qquad k = 0, \ldots, N_z.$$

The $x_{k,\alpha}$ are mole fractions which the DAEs maintain between 0 and - depending on model implementation - 1. Therefore, it is a problem with many states with model induced state bounds.