



## Space splitting convexification: a local solution method for nonconvex optimal control problems

Tadeas Sedlacek, Dirk Odenthal & Dirk Wollherr

To cite this article: Tadeas Sedlacek, Dirk Odenthal & Dirk Wollherr (2021): Space splitting convexification: a local solution method for nonconvex optimal control problems, International Journal of Control, DOI: [10.1080/00207179.2021.2004449](https://doi.org/10.1080/00207179.2021.2004449)

To link to this article: <https://doi.org/10.1080/00207179.2021.2004449>



© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 25 Nov 2021.



Submit your article to this journal [↗](#)



Article views: 232



View related articles [↗](#)



View Crossmark data [↗](#)

# Space splitting convexification: a local solution method for nonconvex optimal control problems

Tadeas Sedlacek <sup>a,b</sup>, Dirk Odenthal <sup>b</sup> and Dirk Wollherr <sup>a</sup>

<sup>a</sup>Chair of Automatic Control Engineering, Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany;

<sup>b</sup>BMW M GmbH, Garching near Munich, Germany

## ABSTRACT

Many optimal control tasks for engineering problems require the solution of nonconvex optimisation problems, which are rather hard to solve. This article presents a novel iterative optimisation procedure for the fast solution of such problems using successive convexification. The approach considers two types of nonconvexities. Firstly, equality constraints with possibly multiple univariate nonlinearities, which can arise for nonlinear system dynamics. Secondly, nonconvex sets comprised of convex subsets, which may occur for semi-active actuators. By introducing additional decision variables and constraints, the decision variable space is decomposed into affine segments yielding a convex subproblem which is solved efficiently in an iterative manner. Under certain conditions, the algorithm converges to a local optimum of a scalable, piecewise linear approximation of the original problem. Furthermore, the algorithm tolerates infeasible initial guesses. Using a single-mass oscillator application, the procedure is compared with a nonlinear programming algorithm, and the sensitivity regarding initial guesses is analysed.

## ARTICLE HISTORY

Received 9 May 2021

Accepted 5 November 2021

## KEYWORDS

Optimal control; successive convexification; collocation; local solution method; exact penalty function

## 1. Introduction

Many optimal control tasks for engineering problems require the solution of nonconvex optimisation problems (OPs), which are harder to solve than convex ones. The nonconvexity can have various causes. Restrictions in the operating range of actuators can result in nonconvex sets. Nonconvexity can also be caused by nonlinear system dynamics resulting in nonlinear and thus nonconvex equality constraints. Since a fast and robust computation of the optimum is always desirable and even essential for real-time control applications, convexification represents a fundamental field in optimal control. Convex OPs can be solved efficiently and robustly with elaborate solvers capable of computing the global optimum with high accuracy and generally short computation times. Various convexification methods have been proposed, which generally solve nonconvex problems by iteratively solving convex substitute problems.

### 1.1 Literature survey on convexification techniques

Lossless convexification (LC) transforms a special class of nonconvex sets into convex ones by lifting the optimisation space into a higher dimension using additional decision variables (Acikmese & Blackmore, 2011; Acikmese & Ploen, 2007). The approach is applicable to a special class of nonconvex sets: sets generated by removing a convex subset from a convex set. This has been used in aeronautic optimal control applications to successfully convexify annulus-shaped sets. An overview regarding the individual findings on LC is given in Raković and Levine (2018, p.338).

Another often employed technique for convex approximation is linearisation since linear objectives and constraints are convex (Boyd & Vandenberghe, 2004). However, linearisation can introduce conservativeness and infeasibility since the resulting accuracy strongly depends on the nonlinearity as well as the linearisation point. Various linearisation techniques have been used to approximate nonlinear systems when applying optimal control methods. In the simplest case, the system is linearised at each time instant around the current operating point, exemplarily implemented in Falcone et al. (2007). However, the model approximation is only sufficiently valid in a close neighbourhood around the linearisation point. With increasing distance to this point, the linearised OP can strongly differ from the original one. Thus, this approach can result in poor controller performance or even lead to infeasibility or instability.

In order to mitigate linearisation errors, the system can be linearised around predicted trajectories resulting in efficiently solvable quadratic programming (QP) problems (Diehl et al., 2005; Seki et al., 2004). One method using iterative linearisation around trajectories in a model predictive control (MPC) setup is the so-called real-time iteration approach (Diehl et al., 2005). The procedure assumes that at every time instant, the shifted solution from the preceding time instant represents a good initial guess close to the actual solution. Thus, a full Newton-step is taken, omitting underlying line-search routines to reduce computation time.

Another iterative scheme linearising around changing trajectories has been presented in Mao et al. (2016). Starting from an initial trajectory, the nonlinear system equations are

successively linearised around the solution computed in the preceding iteration. Adding virtual control inputs eliminates the artificial infeasibility introduced by the linearisation. Furthermore, including trust regions ensures that the solution does not deviate too much from the preceding succession and thus the linearisation represents a valid approximation. This avoids unboundedness of the linearised OP. Based on the ratio between actual objective change and linearised objective change, the algorithm adjusts the trust regions and terminates if the objectives coincide.

A linearise-and-project (LnP) approach has been used in Mao et al. (2017) to resolve the nonconvexity due to nonlinear system equations and nonconvex constraints. The approach presumes convex functions for the right-hand side of the system differential equations. The satisfaction of the system dynamics is ensured by relaxing the corresponding equality constraints into inequality constraints and using exact penalty functions. In an iterative optimisation routine, the computed solution is projected onto the constraints in each iteration to gain adequate linearisation points.

Instead of linearising the system equations around trajectories, a linear input-to-state behaviour can be enforced via feedback-linearisation (Khalil, 2002, p. 505). An optimal control method can then be used to optimally control the linearised system in an outer control loop (Del Re et al., 1993). However, it is possible that the nonlinear mapping of the feedback linearisation transforms originally convex objective and constraint functions into nonconvex ones (Simon et al., 2013). Then, the exact satisfaction of these constraints requires a nonlinear programming (NLP) strategy or an iterative scheme which both eliminate the computational benefits of the feedback-linearisation (Kurtz & Henson, 1997). One possible solution to this problem is approximating these constraints by estimating future inputs based on the preceding time instant within an MPC scheme (Kurtz & Henson, 1997; Schnelle & Eberhard, 2015). The authors in Simon et al. (2013) have proposed an alternative MPC approach that replaces such nonlinear constraints via dynamically generated local inner polytopic approximations rendering the OP convex.

Fuzzy MPC with models of Takagi-Sugeno (TS) type has been successfully used for nonlinear MPC. The TS modelling procedure enables an accurate approximation of nonlinear systems by using data combined with model knowledge (Tanaka & Wang, 2004). The modelling approach decomposes nonlinear models into several local approximations which are blended together via fuzzy rules. However, this requires the system matrices to be stored, thus occupying more memory. Although TS fuzzy models have been successfully used in nonlinear MPC, computation time can be strongly reduced by linearising the TS models around predicted trajectories (Mollov et al., 2004, 1998). The resulting convex QP problem can be solved efficiently while guaranteeing convergence via a line search mechanism.

Differential dynamic programming initially linearises the system equations around a nominal trajectory. Starting from the terminal state, a backward pass identifies the optimal controller gains of a linear quadratic regulator at each time step using the linearised system. Subsequently, a forward pass computes new nominal trajectories via numerical integration using

the controller gains previously computed in the backward pass. This procedure is repeated until convergence. Iterative linear quadratic regulators are a variant of differential dynamic programming (Mayne, 1973). The main difference is that only first-order derivatives are used for the approximation of the system dynamics via Taylor expansion instead of second-order ones which decreases computation time (Tassa et al., 2012). Constrained versions of this optimisation technique have been presented in Tassa et al. (2014), Xie et al. (2017) and Chen et al. (2017).

Lifting-based MPC methods represent another class of linearisation approaches used for solving optimal control problems (OCPs) with nonlinear systems. By lifting the nonlinear dynamics to a space of higher dimension, the evolution of the system can be represented in a bilinear or linear fashion via Carleman linearisation or the Koopman operator framework, respectively. Representing the nonlinear system equations via bilinear terms enables the analytical computation of sensitivity functions (Armaou & Ataei, 2014) and of solutions (Fang & Armaou, 2015) which speeds up computation time. The Koopman MPC approach employs a linear predictor of higher order, which is identified via data sets, to approximate the nonlinear system (Korda & Mezić, 2018). The resulting larger OCP is condensed by eliminating the state decision variables and then solved for the input decision variables using linear MPC. Hence, this method enables a fast solution of nonlinear OCPs. However, identifying the linear Koopman system involves some effort requiring an adequate selection of basis functions, non-recurrent data sets and the solution of convex OPs (Cibulka et al., 2019). Furthermore, the Koopman MPC approach does not always outperform the standard MPC method using local linearisations (Cibulka et al., 2020). Depending on the dimension of the lifted states, both approaches can require the storage of a great number of offline computed matrices.

Various local methods for solving nonconvex quadratically constrained quadratic programming (QCQP) problems have been summarised in d'Aspremont and Boyd (2003) and Park and Boyd (2017). A two-phase coordinate descent method first reduces the maximum constraint violation trying to find a feasible point. The second phase is restricted to feasible points only and searches in each iteration for a feasible point with a better objective function value. The convex-concave procedure (CCP) is a method for finding a local optimum for difference-of-convex programming problems, thus suitable for QCQP. The nonconvex part of the constraints is linearised rendering the constraints convex. In order to deal with infeasible initial guesses, penalty CCP relaxes the linearised constraints via slack variables and introduces a gradually increasing penalty objective for constraint violations. The alternating directions method of multipliers (ADMM) is a variant of the augmented Lagrangian method. It forms an equivalent OP by using auxiliary decision variables which must satisfy a consensus constraint. The objective function is augmented using switching indicator functions which penalise individual constraint violations. The augmented objective function terms are separated into two groups of decision variables. Instead of solving the proximal augmented Lagrangian function for both groups of decision variables simultaneously, the solution is

computed using an alternating approach. First, the first group of decision variables is fixed and the problem is solved for the second group. Then, the solution of the second group is fixed and the problem is solved for the first one. The results are used for the dual variable update and the procedure is repeated. This alternating approach requires the solution of simplified QCQP problems enhancing computation time compared to simultaneously solving for both groups of decision variables.

Convexification measures for sequential quadratic programming (SQP) approaches on a more algorithmic level have been presented in Verschueren (2018). Many QP solvers do not support an indefinite Hessian of the Lagrangian since then a descent direction is not guaranteed. The author has presented a structure-preserving convexification procedure for indefinite Hessians. The convexified Hessian can be fed to any structure-exploiting QP solver (Verschueren, 2018, p. 39). Furthermore, sequential convex quadratic programming is proposed which uses second-order derivatives of convex objective functions and convex constraint functions to construct positive definite Hessians enabling the sequential solution of convex QP problems (Verschueren, 2018, p. 70).

Using the notion of space decomposition for convexification, global optimisation (GO) techniques have similarities with the convexification approach presented in this article. In order to compute the global optimum of continuous, nonconvex OPs, spatial branch-and-bound (BnB) techniques iteratively divide the search space into increasingly smaller subdomains (Liberti, 2008; Liberti & Maculan, 2006; Ryoo & Sahinidis, 1995; Tawarmalani & Sahinidis, 2002). A convex relaxation of nonconvex functions is applied on each subdomain. The efficiently solvable relaxed problem provides a lower bound of the objective function for the corresponding subdomain. The comparison with an upper bound, which can be computed using local optimisation techniques, determines if a further space subdivision is necessary. Thus, this technique searches over a tree whose nodes correspond to individual relaxed problems which consider different subdomains. Comparing the solutions on the individual subdomains provides the global solution. Tree nodes are excluded on the fly based on the evaluation of the lower and upper bounds of the individual nodes eliminating the need to explore the entire domain.

## 1.2 Contribution

In this article, a novel successive convexification method called space splitting convexification (SSC) is proposed aiming at reducing the computation time required to solve nonconvex OPs. The key contribution of the approach is the space decomposition procedure which provides a piecewise linear approximation of the original, nonconvex problem by introducing auxiliary decision variables and absolute value constraints (AVCs). These AVCs possess a beneficial structure for linearisation and point projection onto constraints. The AVCs are transferred to the objective function using the concept of exact penalty functions and then iteratively linearised around changing points. The linearisation of the AVCs results in a binary decision: wrong or right sign of the AVC-linearisation. In each iteration, the computed solution is projected onto the

AVCs and the sign is corrected if necessary enabling the linearisation errors to vanish in subsequent iterations. This concludes the iterative solution of a convex QP problem or even linear programming (LP) problem if the original objective is linear. The violation of the linearised AVCs serves as feedback if the correct sign was chosen which is used as an indicator for convergence. The approach is capable of considering two types of nonconvexities. Firstly, a class of nonconvex sets that can be split into convex subsets which are called zonally convex sets (ZCSs) in this article. Such sets arise for instance when semi-active actuators are used. Secondly, equality constraints with possibly multiple but univariate nonlinearities. As already mentioned, a common source for these nonlinear equality constraints are nonlinear system equations when applying optimal control methods. Requiring only simple mathematical operations, the AVC-projection is of linear computational complexity. Furthermore, the binary nature of the AVC-linearisation generally results in few superordinate iterations. Thus, the SSC algorithm greatly reduces computation time enabling an efficient solution in polynomial-time. Furthermore, robust initialisation properties are given since the initial guess is not required to be feasible. The algorithm converges under certain conditions to local optima of the piecewise linear substitute problem, which represents a scalable approximation of the original problem. However, being a local solution method, the computed local solution generally depends on the provided initial guess. For good initialisations, the algorithm computes solutions which are close to the global optimum.

## 1.3 Article structure

The article is organised as follows. The novel successive convexification method is presented in Section 2. Starting with the general formulation of optimal control problems, the original static OP for the numerical solution of the optimal control task is presented in Section 2.1. Afterwards, Section 2.2 derives the successive convexification algorithm and provides a proof of convergence. The specific application of the SSC algorithm to two-dimensional ZCSs and nonlinear equality constraints is illustrated in Section 2.3. Remarks on the computation time of the algorithm are given in Section 2.4. The comparison of the SSC approach with existing methods in Section 2.5 concludes the theoretical part of the article. Using a single mass oscillator (SMO) application as an example, the SSC algorithm is compared with the NLP solver *IPOPT* (Wächter & Biegler, 2006) in Section 3. The required NLP problem and QP problem are defined in Section 3.1 and the corresponding results are discussed in Section 3.2. Furthermore, the sensitivity regarding initial guesses is analysed in Section 3.3 and rotated space splitting is briefly addressed in Section 3.4. The article closes with concluding remarks in Section 4. In order to promote swift comprehension, a table of notation is included at the end of the article.

## 2. Space splitting convexification

Optimal control deals with computing the optimal input and state trajectories for a given system model. The motion of a nonlinear, continuous, time-invariant system is given by the

differential equation system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad (1)$$

with states  $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$  and inputs  $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ . An optimal control trajectory  $\mathbf{u}(t)$  for a system described by (1) and time interval  $t \in [t_0, t_f]$  can be computed by solving a dynamic OP of following form:

$$\min_{\mathbf{u}(t)} \tilde{J} = \vartheta(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} \phi(\mathbf{x}(t), \mathbf{u}(t), t) dt \quad (2a)$$

$$\text{s.t. } \dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{0} \quad \forall t \in [t_0, t_f], \quad (2b)$$

$$\tilde{\mathbf{c}}(\mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} \quad \forall t \in [t_0, t_f], \quad (2c)$$

$$\tilde{\mathbf{h}}(\mathbf{x}(t), \mathbf{u}(t), t) \leq \mathbf{0} \quad \forall t \in [t_0, t_f]. \quad (2d)$$

The performance index  $\tilde{J}$  in (2a) to be minimised is comprised of the terminal cost function  $\vartheta$  and the intermediate cost term  $\phi$ . Equality constraint (2b) ensures that the system equations (1) are satisfied. General equality constraints  $\tilde{\mathbf{c}} \in \mathbb{R}^{n_c}$  and inequality constraints  $\tilde{\mathbf{h}} \in \mathbb{R}^{n_h}$  are included via (2c) and (2d), respectively. Solving the dynamic OP (2) analytically is generally only possible for simple problems. Thus, dynamic OPs are mostly solved numerically via direct or indirect methods (Sedlacek et al., 2020c). While direct methods solve the original OP, indirect methods deduce a boundary-value problem which is analytically derived from the optimality conditions. A discretisation of trajectories via collocation or shooting can be used to obtain a static OP. Collocation methods employ decision variables for the inputs and states at discrete time points and deduce the trajectory via polynomial interpolation between these points. Single shooting methods only employ decision variables for the inputs and compute the corresponding state trajectories via numerical integration. Multiple shooting methods divide the time interval into subintervals on which the numerical integration is performed. The states at the interval margins represent additional decision variables and continuity constraints are introduced to ensure that the boundary points of the individual integration segments coincide. A direct collocation method is employed in this article due to the simpler initialisation of direct methods and the increased sparsity of collocation methods.

## 2.1 Problem formulation

Using separated Hermite-Simpson collocation with a discretisation into  $n_{\text{seg}}$  segments yields  $n_{\text{pts}} = 2n_{\text{seg}} + 1$  collocation points and thus the vector of decision variables

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_0^T & \mathbf{w}_{\frac{1}{2}}^T & \dots & \mathbf{w}_{n_{\text{seg}}}^T \end{bmatrix}^T \in \mathbb{R}^{n_w}, \mathbf{w}_k := \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \end{bmatrix}. \quad (3)$$

The SSC procedure is applicable to NLP problems of the form

$$\min_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{P}_0 \mathbf{w} + \mathbf{q}_0^T \mathbf{w} + r_0 \quad (4a)$$

$$\text{s.t. } \mathbf{c}_{\text{coll},i}(\mathbf{w}_i, \mathbf{w}_{i+\frac{1}{2}}, \mathbf{w}_{i+1}) = \mathbf{0} \quad \forall i \in \mathcal{I}, \quad (4b)$$

$$\mathbf{c}_k = \mathbf{A}_c \mathbf{w}_k + \mathbf{k}_c = \mathbf{0} \quad \forall k \in \mathcal{K}, \quad (4c)$$

$$\mathbf{h}_k(\mathbf{w}_k) \leq \mathbf{0} \quad \forall k \in \mathcal{K}, \quad (4d)$$

$$\mathbf{h}_{\text{zcs},k}(\mathbf{w}_k) \leq \mathbf{0} \quad \forall k \in \mathcal{K} \quad (4e)$$

with indices  $i \in \mathcal{I} := \{i = 0, 1, 2, \dots, n_{\text{seg}} - 1\}$  and  $k \in \mathcal{K} := \{0, \frac{1}{2}, 1, \dots, n_{\text{seg}}\}$  representing the corresponding collocation segment and collocation point, respectively. Objective (4a) is assumed to be quadratic and convex in the decision variables with  $r_0 \in \mathbb{R}$ ,  $\mathbf{q}_0 \in \mathbb{R}^{n_w}$  and  $0 \leq \mathbf{P}_0 \in \mathbb{R}^{n_w \times n_w}$ . The positive semi-definiteness of  $\mathbf{P}_0$  concludes the convexity of the objective. Collocation constraints (4b) ensure that the system equations (1) are satisfied. For Hermite-Simpson collocation, these equality constraints are given by

$$\mathbf{c}_{\text{coll},i} := \begin{bmatrix} \mathbf{x}_{i+1} - \mathbf{x}_i - \frac{1}{6} \Delta_i (\mathbf{f}_i + 4\mathbf{f}_{i+\frac{1}{2}} + \mathbf{f}_{i+1}) \\ \mathbf{x}_{i+\frac{1}{2}} - \frac{1}{2} (\mathbf{x}_i + \mathbf{x}_{i+1}) - \frac{1}{8} \Delta_i (\mathbf{f}_i - \mathbf{f}_{i+1}) \end{bmatrix} = \mathbf{0} \quad (5)$$

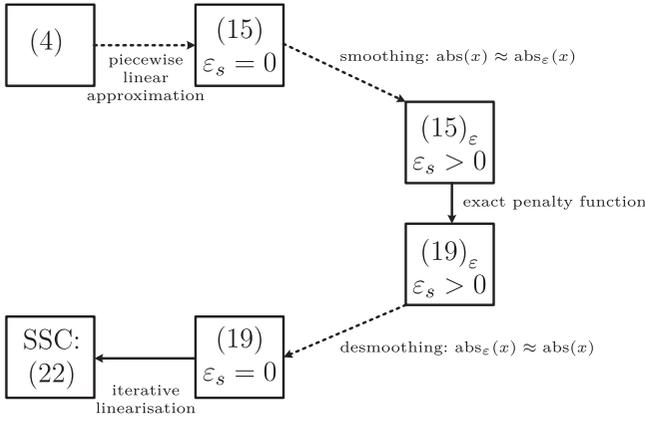
with  $\mathbf{f}_j := \mathbf{f}(\mathbf{x}_j, \mathbf{u}_j)$  and segment width  $\Delta_i := t_{i+1} - t_i$  (Betts, 2010; Kelly, 2017). The SSC approach considers nonlinear right-hand sides of the form

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{k}) + \mathbf{f}_{\text{pwl}}(\mathbf{x}, \mathbf{u}) \quad (6)$$

with  $\mathbf{A} \in \mathbb{R}^{n_x \times n_x}$ ,  $\mathbf{B} \in \mathbb{R}^{n_x \times n_u}$  and  $\mathbf{k}, \mathbf{f}_{\text{pwl}} \in \mathbb{R}^{n_x}$ . Therein  $\mathbf{f}_{\text{pwl}}$  represents a composition of univariate nonlinearities that can be depicted or approximated by piecewise linear curves. For simplicity, it is assumed that the remaining equality constraints (4c) are affine in the decision variables. The inequality constraints (4d) are required to be convex in the decision variables. However, aiming at constructing a QP problem, it is assumed below that the inequality constraints are even affine. This stricter assumption can be posed without loss of generality, since convex inequality constraints can be approximated via a polyhedron using multiple affine functions (Boyd & Vandenberghe, 2004, p. 32). The inequalities (4e) depict ZCSs, which are nonconvex sets that can be split into convex subsets.

## 2.2 Successive convexification procedure

Considering the standard form of convex OPs (Boyd & Vandenberghe, 2004, p. 136), all equality constraints must be affine in the decision variables to enable the convexity of the problem. Hence, a nonlinear right-hand side of the system equations (1) results in nonlinear equality constraints (5), yielding a nonconvex OP. Furthermore, the nonconvex inequality constraints (4e) also prohibit a convex problem. The SSC algorithm iteratively solves a convex substitute problem which is derived from the original problem (4) as schematically illustrated in Figure 1. The core idea of this concept is considering the transformed problem (15) which represents a piecewise linear approximation of the original problem (4). This transformation is achieved by introducing space splitting constraints. These constraints are nonconvex but render the original constraints convex. Furthermore, they possess an advantageous structure which is exploited by the algorithm. In order to deal with the remaining nonconvexity, the nonconvex space splitting constraints are transferred to the objective following the theory of exact penalty functions. The resulting problem (19) possesses a convex feasible set but a nonconvex objective. Thus, the nonconvex part of the objective is iteratively linearised around points that are adjusted based on the previous iterate using an intermediate projection routine. This yields the convex problem (22), which is sequentially



**Figure 1.** Transformation of OP; Dotted and solid arrows represent approximated and exact transformations, respectively.

solved within the SSC algorithm. A smoothable absolute value function is introduced

$$\text{abs}_\varepsilon(x) := \sqrt{x^2 + \varepsilon_s} \approx |x| = \text{abs}(x) \quad \text{with } 0 \leq \varepsilon_s \ll 1 \quad (7)$$

which recovers the true absolute value for  $\varepsilon_s = 0$ . The intermediate OPs  $(15)_\varepsilon$  and  $(19)_\varepsilon$  depicted in Figure 1 apply  $\varepsilon_s > 0$  to provide continuous differentiability for the validity of the optimality conditions legitimising the application of exact penalty functions. Afterwards, this smoothing is removed since it represents an approximation which is not required in problem (22) for continuous differentiability. The subsequent sections describe the problem transformation process in more detail.

### 2.2.1 Space splitting constraints

Before presenting the individual OPs, the basic principle for the space splitting of a decision variable is illustrated in this section providing graphical support. If collocation is used, the inputs as well as states represent decision variables. Thus, the input space and the state space can be split. The splitting procedure is presented using  $w \in \mathcal{W} := [\underline{w}, \bar{w}]$  as a representative decision variable which is split at the transition point  $w = w_{\text{tr}}$ . As illustrated in Figure 2(a), the auxiliary decision variables

$w_{\text{low}} \in \mathcal{W}_{\text{low}} := [w, w_{\text{tr}}]$  and  $w_{\text{up}} \in \mathcal{W}_{\text{up}} := [w_{\text{tr}}, \bar{w}]$  are introduced aiming at satisfying the following relations:

$$w_{\text{low}} = \begin{cases} w & \forall w \leq w_{\text{tr}}, \\ w_{\text{tr}} & \text{else} \end{cases}, \quad w_{\text{up}} = \begin{cases} w & \forall w \geq w_{\text{tr}}, \\ w_{\text{tr}} & \text{else} \end{cases}. \quad (8)$$

**Lemma 2.1:** The splitting relations (8) can be implemented via the splitting constraints

$$g_{\text{aff}} = w_{\text{up}} + w_{\text{low}} - (w + w_{\text{tr}}) = 0 \quad (9a)$$

$$g_{\text{abs}} = \underbrace{w_{\text{up}} - w_{\text{low}}}_{=: w_\Delta} - \underbrace{|w - w_{\text{tr}}|}_{=: \tilde{w}} = 0. \quad (9b)$$

**Proof:** Solving (9a) for one of the auxiliary decision variables yields

$$w_{\text{low}} = w + w_{\text{tr}} - w_{\text{up}}. \quad (10a)$$

Inserting (10a) into (9b) results in

$$w_{\text{up}} = \frac{1}{2} (|w - w_{\text{tr}}| + w + w_{\text{tr}}) \\ \Rightarrow \begin{cases} w_{\text{up}} = w \stackrel{(10a)}{\Rightarrow} w_{\text{low}} = w_{\text{tr}} & \forall w \geq w_{\text{tr}}, \\ w_{\text{up}} = w_{\text{tr}} \stackrel{(10a)}{\Rightarrow} w_{\text{low}} = w & \forall w \leq w_{\text{tr}} \end{cases}. \quad (10b)$$

Thus, constraints (9) ensure the desired splitting according to (8). ■

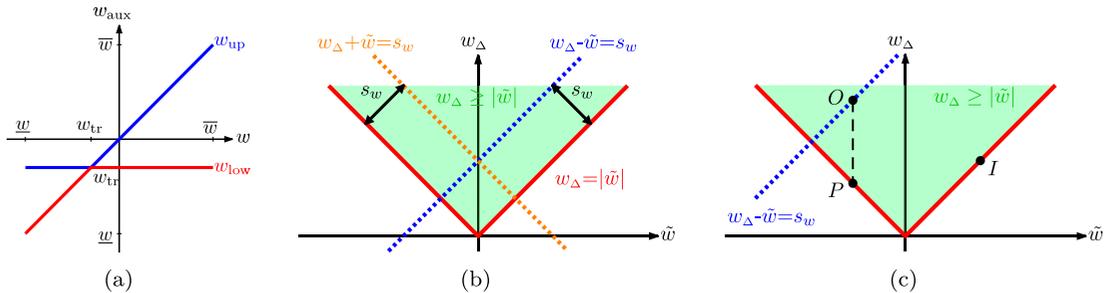
The AVC (9b) is relaxed into a convex inequality constraint which can be depicted using two linear inequality constraints

$$g_{\text{abs}} = w_\Delta - |\tilde{w}| \geq 0 \quad \Leftrightarrow \quad \mathbf{h}_{\text{abs}} = \begin{bmatrix} w_\Delta - \tilde{w} \\ w_\Delta + \tilde{w} \end{bmatrix} \geq \mathbf{0}. \quad (11a)$$

Furthermore, the linearisation of the AVC based on the result of the preceding iteration  $w_{\text{prev}}^*$  is added as an additional objective term in form of an exact penalty function

$$l_{\text{abs}} = w_\Delta - \sigma_w \tilde{w} = 0, \quad \sigma_w := \text{sign}(\tilde{w}^*) = \text{sign}(w_{\text{prev}}^* - w_{\text{tr}}) \quad (11b)$$

$$J_g := \tau l_{\text{abs}} \quad (11c)$$



**Figure 2.** Space splitting. (a) Splitting of decision variable  $w$  into  $w_{\text{low}}$  and  $w_{\text{up}}$ . (b) Absolute value constraint (9b) and its relaxed linearisation according to (13a). (c) Projection after optimisation with initially wrong sign;  $I$ : initial guess;  $O$ : optimum;  $P$ : projection of optimum.

with penalty parameter  $\tau$ . Considering (11b), the linearisation breaks down to choosing the sign  $\sigma_w$  of the absolute value based on the previous solution  $w_{\text{prev}}^*$ . Furthermore, the projection of the previous solution onto the constraints in (9) follows from (8) and provides the initial solution, marked as  $(\cdot)$ , for the subsequent iteration:

$$\check{w} = w_{\text{prev}}^*, \check{w}_{\text{low}} = \min(w_{\text{prev}}^*, w_{\text{tr}}), \check{w}_{\text{up}} = \max(w_{\text{prev}}^*, w_{\text{tr}}). \quad (12)$$

The projection step is depicted in Figure 2(c) for an initial guess with a different sign than the computed solution:  $\text{sign}(\check{w}_{\text{prev}}) \neq \text{sign}(w_{\text{prev}}^*)$ . For a graphical interpretation, the additional cost term (11c) can be replaced by an additional relaxed equality constraint with the slack variable  $s_w \geq 0$  being minimised via a penalty objective:

$$w_{\Delta} - \sigma_w \check{w} = s_w \quad (13a)$$

$$J_s := \tau s_w. \quad (13b)$$

Since adding slack variables increases the number of decision variables, this is not desirable from an implementation standpoint; however, it fosters comprehension. The relaxation of constraint (13a) via the slack variable  $s_w$  is necessary to avoid infeasibility, which is illustrated in Figure 2(b): Since  $g_{\text{abs}} = w_{\Delta} - |\check{w}| \geq 0$  holds, an unrelaxed constraint (13a) with  $s_w \equiv 0$  would only allow values satisfying  $\check{w} \leq 0$  for  $\sigma_w = -1$  and only values  $\check{w} \geq 0$  for  $\sigma_w = 1$ . However, the sign is based on the previous iteration and can thus be wrong. Then, a relaxation represented by the slack variable is necessary to allow values  $\check{w}$  of the opposite region and avoid infeasibility. The sign is corrected for the subsequent iteration step to enable eradicating the additional objective (13b) and thus the slack variable. This concludes that the initial solution does not have to represent a feasible solution, which results in robust initialisation behaviour. The relaxation via the slack variable  $s_w$  in (13a) corresponds to the violation of constraint  $l_{\text{abs}}$  via (11c). The fact that the constraint violation will be large in case of a wrong sign is used as feedback for the algorithm. The algorithm will iterate until a feasible point satisfying  $l_{\text{abs}} \approx 0 \approx s_w$  is reached. Then, the additional objective (11c) vanishes.

### 2.2.2 Piecewise linear approximation of optimisation problem

The first step towards a convex OP is the derivation of a piecewise linear approximation of the original problem (4). This is achieved by using the concept of space splitting presented in the previous section. Since multiple decision variables can be split, the formulation of the OP is posed using the augmented vector of decision variables

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_0^T & \mathbf{z}_{\frac{1}{2}}^T & \cdots & \mathbf{z}_{n_{\text{seg}}}^T \end{bmatrix}^T, \mathbf{z}_k := \begin{bmatrix} \mathbf{x}_k \\ \mathbf{u}_k \\ \mathbf{w}_{\text{aux},k} \end{bmatrix} \in \mathbb{R}^{n_z} \quad (14)$$

with the auxiliary decision variables  $\mathbf{w}_{\text{aux},k} \in \mathbb{R}^{n_s}$ . Therein some of the original states and inputs are excluded when they are depicted using affine transformations as illustrated in Section 2.3.1. The corresponding substitute problem is given for

all  $i \in \mathcal{I}$  and  $k \in \mathcal{K}$  by

$$\min_{\mathbf{z}} J(\mathbf{w}) \quad (15a)$$

$$\text{s.t. } \hat{\mathbf{c}}_{\text{coll},i}(\mathbf{z}) = \mathbf{c}_{\text{coll},i}(\mathbf{w} = \Phi(\mathbf{z})), \mathbf{f}_{\text{pwl}} = \Phi_{\text{pwl}} = \mathbf{0}, \quad (15b)$$

$$\hat{\mathbf{c}}_k(\mathbf{z}) = \mathbf{c}_k(\mathbf{w} = \Phi(\mathbf{z})) = \mathbf{0}, \quad (15c)$$

$$\hat{\mathbf{h}}_k(\mathbf{z}) = \mathbf{h}_k(\mathbf{w} = \Phi(\mathbf{z})) \leq \mathbf{0}, \quad (15d)$$

$$\hat{\mathbf{h}}_{\text{zcs},k}(\mathbf{z}) \leq \mathbf{0}, \quad (15e)$$

$$\mathbf{g}_{\text{aff},k}(\mathbf{z}) := \mathbf{E}_{\Sigma,k} \mathbf{w}_{\text{aux},k} - (\mathbf{E}_{z,k} \mathbf{z}_k + \mathbf{w}_{\text{tr}}) = \mathbf{0}, \quad (15f)$$

$$\mathbf{g}_{\text{abs},k}(\mathbf{z}) := \mathbf{E}_{\Delta,k} \mathbf{w}_{\text{aux},k} - \alpha(\underbrace{\mathbf{E}_{z,k} \mathbf{z}_k - \mathbf{w}_{\text{tr}}}_{=\tilde{\mathbf{z}}_k}) = \mathbf{0}. \quad (15g)$$

Therein  $\alpha(\cdot) : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{n_s}$  represents a function which applies the smoothable absolute value function (7) to each vector component individually: No smoothing occurs for  $\varepsilon_s = 0$ . Furthermore, the right-hand side of the system equations (6) is depicted or approximated by the affine expression

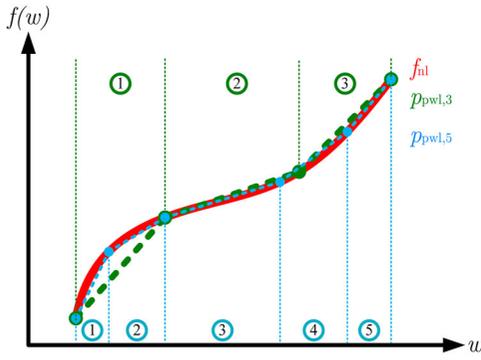
$$\mathbf{f}_{\Phi}(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) = (\mathbf{A} \Phi_x(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) + \mathbf{B} \Phi_u(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) + \mathbf{k}) + \Phi_{\text{pwl}}(\mathbf{x}, \mathbf{u}, \mathbf{w}_{\text{aux}}) \quad (16)$$

with  $\Phi_u : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_u}$  and  $\Phi_x, \Phi_{\text{pwl}} : \mathbb{R}^{n_z} \rightarrow \mathbb{R}^{n_x}$  representing functions which are affine in the decision variables. Thus, (16) enables affine collocation constraints (15b). The transformation function  $\Phi(\mathbf{z})$  in problem (15) represents the accumulation of the affine transformations  $\Phi_x$  and  $\Phi_u$  with  $n_{\Phi} = (n_x + n_u)n_{\text{pts}}$  yielding

$$\mathbf{w} = \Phi(\mathbf{z}) = \begin{bmatrix} \Phi_0 \\ \vdots \\ \Phi_{n_{\text{seg}}} \end{bmatrix} := \hat{\mathbf{A}} \mathbf{z} + \hat{\mathbf{b}}, \Phi_k := \begin{bmatrix} \Phi_x|_k \\ \Phi_u|_k \end{bmatrix} \quad (17)$$

with  $\hat{\mathbf{A}} \in \mathbb{R}^{n_{\Phi} \times n_{\Phi}}$  and  $\hat{\mathbf{b}} \in \mathbb{R}^{n_{\Phi}}$  inserting the affine mapping (17) into the affine constraints (4c) and (4d) results in the affine constraints (15c) and (15d), respectively (Boyd & Vandenberghe, 2004, p. 79). Furthermore, the auxiliary variables are used to represent the ZCSs (4e) using convex functions  $\hat{\mathbf{h}}_{\text{zcs}}$  in (15e). In order to get a QP problem, these constraints are assumed to be affine or approximated by affine inequality constraints. The application of space splitting for the convexification of the original constraints will be shown in more detail in Section 2.3. The splitting constraints (15f) and (15g) represent the constraints (9a) and (9b), respectively. These constraints are necessary to ensure that problem (15) correctly represents the original problem (4). They implement a bisection of selected decision variables at predefined transition values which are stored in the vector  $\mathbf{w}_{\text{tr}}$ . The matrices  $\mathbf{E}_{\Sigma,k}$ ,  $\mathbf{E}_{\Delta,k}$  and  $\mathbf{E}_{z,k}$  are selection matrices extracting the desired decision variables. It is assumed that  $\frac{n_s}{2}$  decision variables are split at each collocation point.

Depending on the individual approximations of the nonlinear constraints, the piecewise linear representation is subject to some error. The approximation error of this substitute problem is defined beforehand by the implemented number of space



**Figure 3.** Approximation of nonlinearity  $f_{nl}$  with three-/five-segmented piecewise linear polynom  $p_{pwl,3}/p_{pwl,5}$ ; Approximation error can be reduced with an increasing number of splitting segments.

subdivisions. If the original problem is piecewise linear, this substitute problem can be an exact representation.

**Remark 2.1:** Let  $f_{nl}(w)$  represent a nonlinearity within a constraint depending on the decision variable representative  $w$ , as depicted in Figure 3. Then, the total absolute error of the piecewise linear approximation  $p_{pwl}(w)$  is given by the sum of absolute errors at each collocation point of the decision variable:

$$e_{pwl} = \sum_{k \in \mathcal{K}} |f_{nl}(w_k) - p_{pwl}(w_k)|. \quad (18)$$

An adequate number and position of knot points for the segmentation into piecewise linear parts achieve sufficiently small approximation errors. However, a higher number of splitting segments increases the size of the OP influencing the computation time.

### 2.2.3 Nonconvex optimisation problem with convex feasible set

The absolute values in equality constraint (15g) still prohibit convexity. Deploying the theory of exact penalty functions (Di Pillo & Grippo, 1989; Han & Mangasarian, 1979), the absolute value equality constraints are moved to the objective without compromising optimality:

**Theorem 2.1:** Let  $\bar{z}$  be a stationary point of the OP

$$\min_{\mathbf{z}} J(\mathbf{w}) + \tau \underbrace{\sum_{k \in \mathcal{K}} \sum_{j=1}^{n_s} g_{abs,k,j}}_{= \|\mathbf{g}_{abs,k}\|_1} \quad (19a)$$

$$\text{s.t.} \quad \begin{bmatrix} \hat{\mathbf{c}}_{coll,i}^T & \hat{\mathbf{c}}_k^T & \mathbf{g}_{aff,k}^T \end{bmatrix} = \mathbf{0}^T \quad \forall i \in \mathcal{I}, k \in \mathcal{K} \quad (19b)$$

$$\begin{bmatrix} \hat{\mathbf{h}}_k^T & \hat{\mathbf{h}}_{zcs,k}^T \end{bmatrix} \leq \mathbf{0}^T \quad \forall k \in \mathcal{K} \quad (19c)$$

$$\mathbf{g}_{abs,k} \geq \mathbf{0} \quad \forall k \in \mathcal{K} \quad (19d)$$

with  $\lambda$  representing the Lagrangian multipliers of the corresponding equality constraints. Since a  $\mathcal{L}_1$ -norm is an exact penalty function,  $\bar{z}$  is a critical point of problem (15) for  $\varepsilon_s > 0$  and sufficiently large but finite penalty weights  $\tau \geq \|\lambda\|_\infty$ .

**Proof:** The prerequisite for the optima recovery is that the optimality conditions are admissible for problem (15) (Nocedal & Wright, 2006, p. 507). For this purpose, the linear independence constraint qualification (LICQ) (Nocedal & Wright, 2006, p. 320) is analysed subsequently. Since the transformation  $\Phi(\mathbf{z})$  in (17) is affine, the gradients of the transformed basic constraints (15c) and (15d) remain linearly independent if the original constraints  $\mathbf{c}(\mathbf{w}) : \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_c}$  and  $\mathbf{h}(\mathbf{w}) : \mathbb{R}^{n_w} \rightarrow \mathbb{R}^{n_h}$  are linearly independent. This can be verified for  $i \in \{1, \dots, n_c\}$  and  $j \in \{1, \dots, n_h\}$  via

$$\nabla_{\mathbf{z}} \hat{c}_i(\mathbf{z}) = \left( \frac{\partial c_i(\Phi(\mathbf{z}))}{\partial \Phi} \frac{\partial \Phi}{\partial \mathbf{z}} \right)^T = \hat{\mathbf{A}}^T \nabla_{\mathbf{w}} c_i(\mathbf{w} = \Phi(\mathbf{z})) \quad (20a)$$

$$\nabla_{\mathbf{z}} \hat{h}_j(\mathbf{z}) = \left( \frac{\partial h_j(\Phi(\mathbf{z}))}{\partial \Phi} \frac{\partial \Phi}{\partial \mathbf{z}} \right)^T = \hat{\mathbf{A}}^T \nabla_{\mathbf{w}} h_j(\mathbf{w} = \Phi(\mathbf{z})). \quad (20b)$$

The gradients of the splitting constraints (15f) and (15g) are by definition linearly independent from each other and from the remaining constraints. Furthermore, it can be assumed without loss of generality that the gradients of the collocation constraints (15b), of the basic constraints (15c)–(15d) and of the ZCS-constraints (15e) are linearly independent. Thus, if the LICQ holds for the original problem (4), then it is also satisfied by the problem (15). By choosing a sufficiently small but positive smoothing parameter  $0 < \varepsilon_s \ll 1$ , the prerequisite of continuous differentiable objective and constraint functions is given while approximation errors are kept small. Thus, the optimal solution will satisfy the Karush–Kuhn–Tucker conditions (Nocedal & Wright, 2006, p. 321). ■

Constraints (19d) are added to enable omitting the norm in the objective as illustrated in (19a). Since  $\mathbf{g}_{abs,k}$  is a vector of concave functions, the AVCs (19d) represent a convex set. Thus, the feasible set of OP (19) is convex.

### 2.2.4 Convex optimisation problem for iterative solution

From Theorem 2.1 follows that solving (19) provides the solution for problem (15) assuming  $\varepsilon_s > 0$ . Unfortunately, the augmented objective (19a) is nonconvex, which will be addressed via linearisation. Subsequently, the smoothing is removed by setting  $\varepsilon_s = 0$  since it will not be required for continuous differentiability. Omitting the smoothing represents an approximation to the smoothed version of (19); however, it is beneficial from a practical standpoint since the splitting constraints are exact for  $\varepsilon_s = 0$ . Furthermore, it enables representing the AVCs (19d) by two affine inequalities which yields exclusively affine constraints required for an LP and QP problem. For a convex OP, the absolute value of the AVCs in the augmented objective is linearised around the points  $\tilde{\mathbf{z}}_k^* \in \mathbb{R}^{n_s}$  yielding the linearised constraints

$$\mathbf{l}_{abs,k}(\mathbf{z}, \tilde{\mathbf{z}}_k^*) := \mathbf{E}_{\Delta,k} \mathbf{w}_{aux,k} - \Sigma_k \tilde{\mathbf{z}}_k = \mathbf{0} \quad \forall k \in \mathcal{K}, \quad (21a)$$

which represent (11b), with the linearisation points being stored in matrix

$$\Sigma := \begin{bmatrix} \Sigma_0 & \Sigma_{\frac{1}{2}} & \dots & \Sigma_{n_{seg}} \end{bmatrix} \quad \text{with} \quad (21b)$$

$$\Sigma_k := \text{diag} \left( \text{sign}(\tilde{z}_{k,1}^*), \dots, \text{sign}(\tilde{z}_{k,n_s}^*) \right).$$

Comparing (15g) and (21a) shows that the linearisation breaks down to a binary decision: choosing the correct sign of the absolute value term  $\tilde{z}_k$ . The linearisation (21a) is affine in the decision variables. Since the sum of convex functions preserves convexity (Boyd & Vandenberghe, 2004, p. 79), the composite objective function is then convex in the decision variables. With all constraints being affine, using linearisation (21a) results in the convex QP problem

$$\min_{\mathbf{z}} J_{\text{tot}}(\mathbf{z}, \mathbf{z}^*) := J(\mathbf{w}) + \tau \sum_{k \in \mathcal{K}} \sum_{j=1}^{n_s} l_{\text{abs},k,j} \quad (22a)$$

$$\text{s.t. } \hat{\mathbf{c}}_{\text{coll},i} = \mathbf{0} \quad \forall i \in \mathcal{I}, \quad (22b)$$

$$\mathbf{c}_{\text{tot},k}^T(\mathbf{z}) := \begin{bmatrix} \hat{\mathbf{c}}_k^T & \mathbf{g}_{\text{aff},k}^T \end{bmatrix} = \mathbf{0}^T \quad \forall k \in \mathcal{K}, \quad (22c)$$

$$\mathbf{h}_{\text{tot},k}^T(\mathbf{z}) := \begin{bmatrix} \hat{\mathbf{h}}_k^T & \hat{\mathbf{h}}_{\text{zcs},k}^T \end{bmatrix} \leq \mathbf{0}^T \quad \forall k \in \mathcal{K}, \quad (22d)$$

$$\mathbf{h}_{\text{abs},k}(\mathbf{z}) := \begin{bmatrix} \mathbf{E}_{\Delta,k} \mathbf{w}_{\text{aux},k} - \tilde{z}_k \\ \mathbf{E}_{\Delta,k} \mathbf{w}_{\text{aux},k} + \tilde{z}_k \end{bmatrix} \geq \mathbf{0} \quad \forall k \in \mathcal{K}. \quad (22e)$$

For linear objectives  $J(\mathbf{w})$ , the SSC approach results in a LP problem. The convex problem (22) is iteratively solved, whereas the linearisation points in  $\Sigma$  are updated based on the solution  $\mathbf{z}^{*\text{T}} = [\mathbf{z}_0^{*\text{T}} \dots \mathbf{z}_{n_{\text{seg}}}^{*\text{T}}]$  of the preceding iteration yielding

$$\tilde{\mathbf{z}}_k^* = \mathbf{E}_{z,k} \mathbf{z}_k^* - \mathbf{w}_{\text{tr}} \quad \forall k \in \mathcal{K}. \quad (23)$$

However, the question arises if replacing the constraint  $\mathbf{g}_{\text{abs}}$  in the objective with its linearisation  $\mathbf{l}_{\text{abs}}$  falsifies the solution. This motivates the following lemma:

**Lemma 2.2:** *The satisfaction of  $\mathbf{h}_{\text{abs},k}(\mathbf{z}) \geq \mathbf{0}$  and  $\mathbf{l}_{\text{abs},k}(\mathbf{z}, \tilde{\mathbf{z}}_k^*) = \mathbf{0}$  concludes  $\mathbf{g}_{\text{abs},k}(\mathbf{z}) = \mathbf{0}$ , independently of the linearisation point  $\tilde{\mathbf{z}}_k^*$ .*

**Proof:** This can be verified using Figure 2(b) as graphical support. For this purpose, the following equations are given in general notation and are compared with the equations from the illustrative example in Section 2.2.1 using the vector of decision variables  $\mathbf{z}_{\text{ex}} = [w_{\text{up}} \ w_{\text{ow}}]^T$ . The nonsmoothed absolute value constraint  $\mathbf{g}_{\text{abs},k,j}$  spans the following set of points:

$$\begin{aligned} \mathcal{G}_{k,j}(\mathbf{z}) &:= \{\mathbf{z} \mid \mathbf{e}_{\Delta,k}^T \mathbf{w}_{\text{aux},k} - |\tilde{z}_{k,j}| = 0\} \\ &\hat{=} \{\mathbf{z}_{\text{ex}} \mid w_{\Delta} - |\tilde{w}| = 0\} \end{aligned} \quad (24a)$$

with  $\mathbf{e}_{\Delta,k}^T$  representing the  $j$ th row of matrix  $\mathbf{E}_{\Delta,k}$ . The admissible points for the AVC-linearisation  $\mathbf{l}_{\text{abs},k,j}$  are given by

$$\begin{aligned} \mathcal{L}_{k,j}(\mathbf{z}, \tilde{\mathbf{z}}_{k,j}^*) &:= \{\mathbf{z} \mid \mathbf{e}_{\Delta,k}^T \mathbf{w}_{\text{aux},k} - \text{sign}(\tilde{z}_{k,j}^*) \tilde{z}_{k,j} = 0\} \\ &\hat{=} \{\mathbf{z}_{\text{ex}} \mid w_{\Delta} - \sigma_w \tilde{w} = 0\} \end{aligned} \quad (24b)$$

representing the infinite extension of the positive and negative branch of the AVC for  $\tilde{z}_{k,j}^*, \sigma_w \geq 0$  and  $\tilde{z}_{k,j}^*, \sigma_w \leq 0$ , respectively. Then, the intersection with the admissible points of the inequality constraint  $\mathbf{h}_{\text{abs},k,j}$

$$\begin{aligned} \mathcal{H}_{k,j}(\mathbf{z}) &:= \left\{ \mathbf{z} \mid \begin{bmatrix} \mathbf{e}_{\Delta,k}^T \mathbf{w}_{\text{aux},k} - \tilde{z}_{k,j} \\ \mathbf{e}_{\Delta,k}^T \mathbf{w}_{\text{aux},k} + \tilde{z}_{k,j} \end{bmatrix} \geq \mathbf{0} \right\} \\ &\hat{=} \left\{ \mathbf{z}_{\text{ex}} \mid \begin{bmatrix} w_{\Delta} - \tilde{w} \\ w_{\Delta} + \tilde{w} \end{bmatrix} \geq \mathbf{0} \right\} \end{aligned} \quad (24c)$$

provides the sets

$$\mathcal{N}_{k,j}(\mathbf{z}) := \{\mathbf{z} \mid \mathbf{z} \in \mathcal{H}_{k,j}, \mathbf{z} \in \mathcal{L}_{k,j}(\mathbf{z}, \tilde{\mathbf{z}}_{k,j}^*) \mid \tilde{z}_{k,j}^* \leq 0\} \quad (24d)$$

$$\mathcal{P}_{k,j}(\mathbf{z}) := \{\mathbf{z} \mid \mathbf{z} \in \mathcal{H}_{k,j}, \mathbf{z} \in \mathcal{L}_{k,j}(\mathbf{z}, \tilde{\mathbf{z}}_{k,j}^*) \mid \tilde{z}_{k,j}^* \geq 0\}, \quad (24e)$$

which represent the negative and positive branch of the AVC and are thus a subset:  $\mathcal{N}_{k,j}, \mathcal{P}_{k,j} \subset \mathcal{G}_{k,j}$ . ■

This important lemma ensures that convergence of problem (22) to a solution which satisfies the linearised constraints  $\mathbf{l}_{\text{abs},k}(\mathbf{z}, \tilde{\mathbf{z}}_k^*) = \mathbf{0}$  also fulfils the original nonlinear constraints  $\mathbf{g}_{\text{abs},k}(\mathbf{z}) = \mathbf{0}$  in (15g).

### 2.2.5 Space splitting convexification algorithm

Subsequently, the SSC algorithm 1 is presented in detail and the proof of its convergence is provided. The algorithm inputs are described in line 1. An initial solution for the inputs  $\mathbf{U}_0^* \in \mathbb{R}^{n_u \times n_{\text{pts}}}$  and states  $\mathbf{X}_0^* \in \mathbb{R}^{n_x \times n_{\text{pts}}}$  must be provided. The algorithm iteratively solves the convex problem (22) using the solution of the previous iterate to correct the signs of the linearisation if necessary. The algorithm terminates when all signs are chosen correctly or the maximum number of iterations  $q_{\text{max}}$  is reached. Thus, if the algorithm converges before reaching the maximum number of iterations  $q < q_{\text{max}}$ , the largest constraint violation is smaller than a small value  $\bar{l}_{\text{abs}} \leq \varepsilon_g$  with  $0 \leq \varepsilon_g \ll 1$ . As already mentioned, the violation of the linearised AVCs indicates that a wrong sign was chosen. The reason why the constraint violations are a suitable feedback for correct sign selection can be visualised using Figure 2(b). The signs and thus the linearised constraints  $\mathbf{l}_{\text{abs},k,j}$  in the objective (22a) are fixed before optimisation. Then, running into the region of  $\tilde{w} \hat{=} \tilde{z}_{k,j}$  with the opposite sign requires increasing  $w_{\Delta} \hat{=} \mathbf{e}_{\Delta,k}^T \mathbf{w}_{\text{aux},k}$  due to the constraints  $\mathbf{h}_{\text{abs}} \hat{=} \mathbf{h}_{\text{abs},k} \geq \mathbf{0}$ , which in turn increases the linearised constraints in the objective. Based on the solution of the preceding iteration, the updating routine in line 7 adjusts the convex QP problem (22) to satisfy (15f)–(15g) using the projection procedure (12): The update routine computes the projected initial solution  $\tilde{\mathbf{Z}}$  and the correct

---

**Algorithm 1** Space Splitting Convexification: a procedure for iterative optimisation.

---

1: **INPUTS:**

- (1)  $\mathbf{U}_0^*, \mathbf{X}_0^*$ : initial solution for input and state decision variables
- (2)  $q_{\text{max}}$ : maximum number of iterations
- (3)  $\varepsilon_g$ : error tolerance for linearised absolute value constraints (AVC)
- (4)  $\underline{\tau}, \bar{\tau}$ : weight bounds for penalty objectives

2: **OUTPUT:**  $\mathbf{U}^*, \mathbf{X}^*$ : system solution within tolerance  $\varepsilon_g$

3: **MAIN FUNCTION:**

4:  $q = 1, \bar{l}_{\text{abs}} = \infty$

5: **while**  $q \leq q_{\text{max}}$  and  $\bar{l}_{\text{abs}} > \varepsilon_g$  **do**

6:    $\tau = \frac{\bar{\tau} - \underline{\tau}}{q_{\text{max}}} \cdot q + \underline{\tau}$  // weight for additional objectives

7:    $[\Sigma, \tilde{\mathbf{Z}}] = \text{updateQP}(\mathbf{U}_{q-1}^*, \mathbf{X}_{q-1}^*)$  // linearisation and initialisation

8:    $[\mathbf{U}_q^*, \mathbf{X}_q^*] = \text{solveQP}(\tau, \Sigma, \tilde{\mathbf{Z}})$

9:    $\bar{l}_{\text{abs}} = \left\| \begin{bmatrix} \mathbf{l}_{\text{abs},0} & \mathbf{l}_{\text{abs},\frac{1}{2}} & \dots & \mathbf{l}_{\text{abs},n_{\text{seg}}} \end{bmatrix} \right\|_{\text{max}}$  // maximum value of all relaxations

10:    $q = q + 1$

11: **end while**

12:  $\mathbf{U}^* = \mathbf{U}_q^*$   
 $\mathbf{X}^* = \mathbf{X}_q^*$  // output final solution

---

signs  $\Sigma$  for the linearisation of the AVCs. Afterwards, the optimiser is invoked in line 8 to solve the convex QP problem or LP problem if the main objective is linear. The theory of exact penalty functions states that the penalty parameter  $\tau$  must be larger than the largest optimal dual variable of the corresponding constraints. Since estimating the limit value for the penalty parameter is a rather complicated task, Algorithm 1 pursues the straightforward approach in line 6 of gradually increasing the penalty parameter up to a high value. A reasonably high but finite upper bound on the penalty parameter  $\bar{\tau}$  avoids numerical problems. Although this is not the best updating approach, it can work well in practice (Nocedal & Wright, 2006, p. 511) as it is the case for the examples considered in this article.

Finally, the following theorem verifies that the algorithm produces a sequence of solutions which eventually converges to a local solution satisfying the AVCs:

**Theorem 2.2:** *Let the feasible set of problem (22) be denoted with*

$$\Omega_f := \left\{ \mathbf{z} \mid \hat{\mathbf{c}}_{\text{coll},i} = \mathbf{0}, \mathbf{c}_{\text{tot},k}(\mathbf{z}) = \mathbf{0}, \right. \\ \left. \mathbf{h}_{\text{tot},k}(\mathbf{z}) \leq \mathbf{0}, \mathbf{h}_{\text{abs},k}(\mathbf{z}) \geq \mathbf{0}, i \in \mathcal{I}, k \in \mathcal{K} \right\}, \quad (25)$$

which is a convex set. Starting from a point  $\mathbf{z}^{(0)}$ , Algorithm 1 generates a sequence  $\{\mathbf{z}^{(q)}\}$  according to

$$\mathbf{z}^{(q+1)} = \min_{\mathbf{z}} \left\{ J_{\text{tot}}(\mathbf{z}, \mathbf{z}^{(q)}) \mid \mathbf{z} \in \Omega_f \right\} \quad q = 0, 1, \dots \quad (26)$$

with  $J_{\text{tot}}$  being a convex function. This sequence eventually converges to a limit point  $\mathbf{z}^*$  which satisfies  $\mathbf{g}_{\text{abs},k} = \mathbf{0} \forall k \in \mathcal{K}$ .

**Proof:** Algorithm 1 iteratively solves problem (22) and gradually increases the penalty parameter up to a finite value  $\tau \leq \bar{\tau}$ . Let the condition  $\tau \geq \|\lambda\|_{\infty}$  mentioned in Theorem 2.1 hold for  $q \geq \mathcal{Q}_{\lambda} \in \mathbb{N}$ . Then, the theory of exact penalty functions states that  $\mathbf{I}_{\text{abs},k} = \mathbf{0} \forall k \in \mathcal{K}$  is satisfied for all  $q \geq \mathcal{Q}_{\lambda}$ . As described in Lemma 2.2, this concludes that only solutions on one branch of the AVCs are admissible. Thus, no sign changes of  $\tilde{z}_{k,j} \hat{=} \tilde{w}$  are possible for  $q \geq \mathcal{Q}_{\lambda}$ . This concludes that the linearisation points of  $\mathbf{I}_{\text{abs},k} \forall k \in \mathcal{K}$  and thus OP (22) remain unchanged for  $q \geq \mathcal{Q}_{\lambda}$ . The convexity of the problem entails that only one optimum exists yielding

$$\mathbf{z}^{(q+1)} - \mathbf{z}^{(q)} = \mathbf{0} \Rightarrow \left\| \mathbf{z}^{(q+1)} - \mathbf{z}^{(q)} \right\| = 0 \quad \forall q \geq \mathcal{Q}_{\lambda}. \quad (27)$$

This concludes that for every  $\varepsilon_{\tau} > 0$ , there is a  $\mathcal{Q} \in \mathbb{N}$  such that

$$\|\Delta \mathbf{z}\| := \left\| \mathbf{z}^{(n)} - \mathbf{z}^{(m)} \right\| < \varepsilon_{\tau} \quad \text{for every } n, m \geq \mathcal{Q} \quad (28)$$

since (28) holds at the latest from the value  $\mathcal{Q} = \mathcal{Q}_{\lambda}$  with  $\|\Delta \mathbf{z}\| = 0 < \varepsilon_{\tau}$ . This proves that (26) is a Cauchy sequence and thus converges (Zorich, 2016, p. 85). Hence, the solutions of the individual iterations can oscillate; however, this oscillation vanishes with an increasing number of iterations finally converging to a single point  $\mathbf{z}^*$ . Due to Lemma 2.2,  $\mathbf{I}_{\text{abs},k} = \mathbf{0} \forall k \in \mathcal{K}$  concludes that this point satisfies  $\mathbf{g}_{\text{abs},k} = \mathbf{0} \forall k \in \mathcal{K}$ . ■

Theorem 2.2 concludes that the iterative solution of OP (22) within the SSC algorithm basically solves the OP (15) with

$\varepsilon_s = 0$ : From convergence to  $\mathbf{I}_{\text{abs},k} = \mathbf{0} \forall k \in \mathcal{K}$  follows that the computed solution satisfies  $\mathbf{g}_{\text{abs},k} = \mathbf{0} = \mathbf{h}_{\text{abs},k} \forall k \in \mathcal{K}$  and the augmented objective term vanishes. The remaining constraints of problems (15) and (22) are identical. Thus, the SSC algorithm computes a solution that is arbitrarily close, but not necessarily identical, to the solution of the piecewise linearly approximated problem (15). The potential discrepancy in solutions is rooted in the necessity to smoothen the problem (15) with  $\varepsilon_s > 0$  for the admissibility of the optimality conditions in Theorem 2.1. Since the linearisation removes the discontinuity, the smoothing is disregarded within the SSC algorithm. From a practical standpoint, the discrepancy is negligible since the smoothing parameter can be assumed arbitrarily small  $\varepsilon_s \rightarrow 0$  making the impact of the smoothing numerically irrelevant. This procedure for the approximation of discontinuities in optimal control problems is common practice, see e.g. (Perantoni & Limebeer, 2014). In summary, the SSC approach chooses one branch of the discontinuity, resulting in the continuously differentiable problem (22) with  $\varepsilon_s = 0$ , and switches in the following iterations to the other branch if the wrong side of the discontinuity was selected.

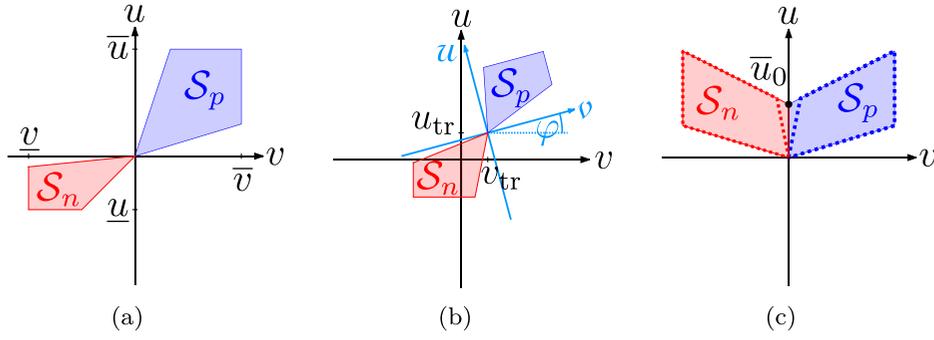
## 2.3 Constraint convexification via space splitting

The previous sections derived the basic structure of the convex OP (22a), which is solved in an iterative manner. However, the question of how the original constraints can be convexified via space splitting to gain the convex constraints (15b)–(15e) is still open. The basic procedure of bisecting the space of a variable into two subdomains has been illustrated in Section 2.2.1. This can be employed to consider ZCSs and equality constraints with univariate nonlinearities in a convex manner, which is demonstrated in Sections 2.3.1 and 2.3.2, respectively. Subsequently, the index for the collocation discretisation  $k \in \mathcal{K}$  is omitted for readability.

### 2.3.1 Convexification of zonally convex sets

One prominent physical example of ZCSs are constraints ensuring semi-activity. Semi-active actuators like limited slip differentials and semi-active dampers are used in many engineering applications due to their good compromise between low energy consumption and high performance (Savarese et al., 2010; Sedlacek et al., 2020a, 2020c). A typical ZCS for a semi-active damper is depicted in Figure 4(a) with damper force  $F_d = u$  and damper velocity  $v$ . The nonconvexity of the set is apparent considering that it is easy to draw a line that contains non-admissible points and connects an admissible point in the first quadrant with an admissible point in the third quadrant. Note that simply linearising the inequality constraints depicting this nonconvex set results in a half-plane, which represents a poor approximation of the set. The main idea for the novel convexification approach is appropriately decomposing nonconvex sets, described in the original decision variables, into convex subsets using auxiliary decision variables. The applicability of the SSC method to the ZCS types depicted in Figure 4 is proven in the subsequent paragraphs.

The approach is initially illustrated using the ZCS depicted in Figure 4(a). Let the convex subset in the first and third quadrant



**Figure 4.** Space splitting convexification of zonally convex sets. (a) ZCS with a single transition point. (b) Shifted, rotated ZCS with a single transition point. (c) ZCS with shared line at transition.

be given by

$$\mathcal{S}_p := \left\{ (u, v) \mid u \in \mathcal{U}, v \in \mathcal{V}, \hat{\mathbf{h}}_{zcs,p}(u, v) \leq \mathbf{0} \right\} \quad \text{and} \quad (29a)$$

$$\mathcal{S}_n := \left\{ (u, v) \mid u \in \mathcal{U}, v \in \mathcal{V}, \hat{\mathbf{h}}_{zcs,n}(u, v) \leq \mathbf{0} \right\}, \quad (29b)$$

respectively. Therein  $v$  and  $u$  are original decision variables. Moreover,  $\hat{\mathbf{h}}_{zcs,p} \leq \mathbf{0}$  and  $\hat{\mathbf{h}}_{zcs,n} \leq \mathbf{0}$  are convex sets. The union of convex sets does not necessarily result in a convex set. As illustrated in Figure 4(a), the union  $\mathcal{S} = \mathcal{S}_p \cup \mathcal{S}_n$  is assumed to be nonconvex in this article. Thus,  $\mathcal{S}$  is a nonconvex set comprised of convex subsets: a ZCS. The velocity  $v$  is split at  $v = v_{tr} = 0$  by a vertical line following Lemma 2.1. The two auxiliary variables  $v_n \in \mathcal{V}_n := [\underline{v}, v_{tr}] = [\underline{v}, 0]$  and  $v_p \in \mathcal{V}_p := [v_{tr}, \bar{v}] = [0, \bar{v}]$  are introduced aiming at implementing

$$v_n = \begin{cases} v & \forall v \leq 0, \\ 0 & \text{else} \end{cases} \quad \text{and} \quad v_p = \begin{cases} v & \forall v \geq 0, \\ 0 & \text{else} \end{cases}. \quad (30)$$

Introducing analogous constraints to (9a) and (11a) as well as additional objective terms in form of (11c) ensures the relations in (30). Furthermore, the auxiliary variables  $u_n \in \mathcal{U}_n := [\underline{u}, u_{tr}]$  and  $u_p \in \mathcal{U}_p := [u_{tr}, \bar{u}]$  are introduced for the individual subsets. Therein  $u_{tr}$  represents the ordinate value of the common point, which lies on the origin in the current example depicted in Figure 4(a) resulting in  $u_{tr} = 0$ . However, the common point of the subsets is not required to lie on the abscissa or ordinate. The vector of auxiliary decision variables is given by  $\mathbf{w}_{aux} = [v_p \ v_n \ u_p \ u_n]$ . While the original state variable  $v$  remains a decision variable in the problem, the input variable is replaced according to the transformation

$$\Phi_u := u_n + u_p - u_{tr} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ u_n \end{bmatrix} - u_{tr}, \quad (31)$$

which is affine in the decision variables. The convex subsets in terms of the auxiliary decision variables are given by

$$\hat{\mathcal{S}}_p := \left\{ (u_p, v_p) \mid u_p \in \mathcal{U}_p, v_p \in \mathcal{V}_p, \hat{\mathbf{h}}_{zcs,p}(u_p, v_p) \leq \mathbf{0} \right\} \quad (32a)$$

$$\hat{\mathcal{S}}_n := \left\{ (u_n, v_n) \mid u_n \in \mathcal{U}_n, v_n \in \mathcal{V}_n, \hat{\mathbf{h}}_{zcs,n}(u_n, v_n) \leq \mathbf{0} \right\}. \quad (32b)$$

Hence, the convex subsets  $\hat{\mathcal{S}}_p$  and  $\hat{\mathcal{S}}_n$  are now defined for separate decision variables and will be linked via the affine mapping (31). This enables the formulation of convex constraints

as depicted in problem (15). The proof that this procedure produces the same set as  $\mathcal{S}$  will be given in Theorem 2.3.

For convexity,  $\hat{\mathbf{h}}_{zcs,p}$  and  $\hat{\mathbf{h}}_{zcs,n}$  in (32a) and (32b) must be convex functions. As exemplarily illustrated in Figure 4(a), the convex subsets are assumed in this article to be representable by  $n_{zcs} = n_{zcs,n} + n_{zcs,p}$  affine inequality constraints

$$\hat{\mathbf{h}}_{zcs,p}(u, v) := \mathbf{A}_{zcs,p} \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b}_{zcs,p} \leq \mathbf{0} \quad (33a)$$

$$\hat{\mathbf{h}}_{zcs,n}(u, v) := \mathbf{A}_{zcs,n} \begin{bmatrix} u \\ v \end{bmatrix} + \mathbf{b}_{zcs,n} \leq \mathbf{0} \quad (33b)$$

with  $\mathbf{A}_{zcs,p} \in \mathbb{R}^{n_{zcs,p} \times 2}$ ,  $\mathbf{b}_{zcs,p} \in \mathbb{R}^{n_{zcs,p}}$ ,  $\mathbf{A}_{zcs,n} \in \mathbb{R}^{n_{zcs,n} \times 2}$  and  $\mathbf{b}_{zcs,n} \in \mathbb{R}^{n_{zcs,n}}$ . These constraints represent (15e) in the convex OP. Although nonlinear, convex functions for  $\hat{\mathbf{h}}_{zcs,p}$  and  $\hat{\mathbf{h}}_{zcs,n}$  would not prevent convexity, introducing such constraints would prohibit an LP or QP problem, which can be solved especially efficiently. Asymmetric subsets like the one illustrated in Figure 4(a) can be easily implemented by using varying coefficients in (33a) and (33b) or by using different domains for  $\mathcal{U}_n$  and  $\mathcal{U}_p$ .

**Theorem 2.3:** *The subsets (32a) and (32b) in combination with the affine mapping (31) depict the set  $\mathcal{S}$  which is given by the union of the convex subsets (29a) and (29b).*

**Proof:** For proving that the splitting approach provides sets that depict the original one, the individual cases for the bisected variable space are considered subsequently.

**Case 1:**  $v_{tr} < v$

The decision variable  $v$  is split at  $v = v_{tr}$ . Thus,  $v_n = v_{tr}$  holds according to Lemma 2.1. This concludes  $u_n = u_{tr}$  since the subsets converge to the common point  $(u_{tr}, v_{tr})$  when leaving the subset towards the other subset. Lemma 2.1 also concludes  $v_p = v$ . Then, the affine mapping (31) is given by  $\Phi_u = u_p \in \hat{\mathcal{S}}_p$ . For  $v > v_{tr}$ ,  $\mathcal{S} = \mathcal{S}_p$  holds. Since  $v_p = v$ ,  $u \in \mathcal{S}_p$  and  $\Phi_u \in \hat{\mathcal{S}}_p$  provide the same feasible set. The chain of conclusions for the remaining cases follows the same reasoning but is given below in equations for conciseness.

**Case 2:**  $v < v_{tr}$

$$\left. \begin{matrix} v_p = v_{tr} \Rightarrow u_p = u_{tr} \\ v_n = v \end{matrix} \right\} \Rightarrow \Phi_u = u_n \in \hat{\mathcal{S}}_n \hat{=} u \in \mathcal{S} = \mathcal{S}_n. \quad (34a)$$

**Case 3:**  $v = v_{tr}$

$$\left. \begin{array}{l} v_p = v_{tr} \Rightarrow u_p = u_{tr} \\ v_n = v_{tr} \Rightarrow u_n = u_{tr} \end{array} \right\} \Rightarrow \Phi_u = u_{tr} \hat{=} u = u_{tr}. \quad (34b)$$

Thus, inserting the affine mapping (31) into the constraints while replacing the original set  $\mathcal{S}$  with the sets (32a) and (32b) yields the same feasible set. ■

The presented proof requires a space splitting of the variable  $v$  which is represented by splitting the convex subsets via a vertical line. Thus, the SSC approach is capable of convexifying two-dimensional ZCSs, which can be divided into two convex subsets connected in a single point using any vertical line. Following theorem is added to show that this also holds for any splitting line which is straight but possibly rotated:

**Theorem 2.4:** *Space splitting applied to a rotated, two-dimensional ZCS also results in convex constraints.*

**Proof:** This can be verified by applying the previously described procedure to a rotated coordinate frame, exemplarily depicted in Figure 4(b). The auxiliary decision variables need to split the rotated two-dimensional space spanned by the variables  $v$  and  $u$ . Assuming the coordinate system is rotated by the angle  $\varphi$  with counter-clockwise rotations depicting positive rotations, following relations hold between the original coordinates  $v-u$  and the rotated coordinates  $v-u$ :

$$\begin{aligned} \begin{bmatrix} v \\ u \end{bmatrix} &= \underbrace{\begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix}}_{=:R(\varphi)} \begin{bmatrix} v - v_{tr} \\ u - u_{tr} \end{bmatrix} \quad \text{and} \\ \begin{bmatrix} v \\ u \end{bmatrix} &= \underbrace{\begin{bmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{bmatrix}}_{=:R(-\varphi)} \begin{bmatrix} v \\ u \end{bmatrix} + \begin{bmatrix} v_{tr} \\ u_{tr} \end{bmatrix}. \end{aligned} \quad (35)$$

The vector of auxiliary decision variables is chosen as  $\mathbf{w}_{aux} = [v_p \ v_n \ u_p \ u_n]$ . For a concise notation, the equations below use the abbreviations  $c_\varphi := \cos(\varphi)$  and  $s_\varphi := \sin(\varphi)$ . Constraints (33) are analogously formulated on the rotated, auxiliary decision variables:  $\hat{\mathbf{h}}_{zcs,p}(u_p, v_p)$  and  $\hat{\mathbf{h}}_{zcs,n}(u_n, v_n)$ . However, an adjustment using (35) is necessary for the constraints (9a), (11a) and (11b) as well as the input mapping (31):

$$g_{aff} = (v_p + v_n) - \tilde{v} = 0 \quad \text{with} \quad (36a)$$

$$\tilde{v} = c_\varphi(v - v_{tr}) + s_\varphi(s_\varphi(v_p + v_n) + c_\varphi(u_p + u_n))$$

$$g_{abs} = (v_p - v_n) - |\tilde{v}| \geq 0 \quad (36b)$$

$$l_{abs} = (v_p - v_n) - \text{sign}(\tilde{v})\tilde{v} \geq 0 \quad (36c)$$

$$\Phi_u = u = \begin{bmatrix} s_\varphi & s_\varphi \end{bmatrix} \begin{bmatrix} v_p \\ v_n \end{bmatrix} + \begin{bmatrix} c_\varphi & c_\varphi \end{bmatrix} \begin{bmatrix} u_p \\ u_n \end{bmatrix} + u_{tr}. \quad (36d)$$

Since the angle  $\varphi$  is constant, the trigonometric functions represent constant values. Thus, a rotated space splitting procedure also yields affine constraints and mappings enabling the formulation of a convex OP. ■

The previous theorems required the subsets to be connected in a single point. Following theorem states that the SSC

approach can be used to approximate ZCSs which share a line of points:

**Theorem 2.5:** *When a two-dimensional ZCSs is comprised of two convex subsets sharing a line of points, the SSC approach can only approximate the original set however with sufficiently small approximation error.*

**Proof:** Without loss of generality, this can be analysed considering Figure 4(c) as example. For the depicted case, the transition line lies on the ordinate  $v_{tr} = 0$ . Analysing the individual cases shows that using the original subsets within the SSC approach would yield a larger set than the original set:

**Case 1:**  $v_{tr} < v$

$$\left. \begin{array}{l} v_p = v \Rightarrow u_p \in \hat{\mathcal{S}}_p \\ v_n = v_{tr} \Rightarrow u_n \in [0, \bar{u}_0] \end{array} \right\} \Rightarrow \Phi_u = (u_n + u_p) \in \hat{\mathcal{S}}_{\Phi,p} \supset \hat{\mathcal{S}}_p \\ \not\cong u \in \mathcal{S} = \mathcal{S}_p. \quad (37a)$$

**Case 2:**  $v < v_{tr}$

$$\left. \begin{array}{l} v_p = v_{tr} \Rightarrow u_p \in [0, \bar{u}_0] \\ v_n = v \Rightarrow u_n \in \hat{\mathcal{S}}_n \end{array} \right\} \Rightarrow \Phi_u = (u_n + u_p) \in \hat{\mathcal{S}}_{\Phi,n} \supset \hat{\mathcal{S}}_n \\ \not\cong u \in \mathcal{S} = \mathcal{S}_n. \quad (37b)$$

**Case 3:**  $v = v_{tr}$

$$\left. \begin{array}{l} v_p = v_{tr} \Rightarrow u_p \in [0, \bar{u}_0] \\ v_n = v_{tr} \Rightarrow u_n \in [0, \bar{u}_0] \end{array} \right\} \Rightarrow \Phi_u = u_n + u_p \in [0, 2\bar{u}_0] \\ \not\cong u \in [0, \bar{u}_0]. \quad (37c)$$

Thus, using the original subsets can render non-admissible points of the original problem admissible. When the input variable is replaced by the auxiliary inputs according to (31), both sets must be adjusted at the transition line  $v = v_{tr}$ . The simplest set adjustment is enforcing  $u_n = 0 = u_p$  at the transition axis, as illustrated in Figure 4 by the dotted lines. Then, the subsets are connected in a single point enabling the application of Theorem 2.4. However, other set manipulations are possible and can be numerically better. Generally, such set approximations introduce conservativeness for the sake of feasible trajectories. However, the introduced error is negligible when the removed area is sufficiently small which can be implemented by choosing steep linear constraints at the transition between the subsets. ■

**Remark 2.2:** In this article, only two-dimensional ZCSs which can be split into two convex subsets have been considered. However, the SSC approach can be applied to certain two-dimensional ZCSs with more than two subsets. For instance, consider three convex subsets positioned horizontally to each other, whereas each subset is connected to the next subset in a single point on the abscissa. This can be verified using the procedures described previously in this section.

This section illustrated the convexification procedure for ZCSs in regards to inputs. This results in the depicted affine input transformation  $\Phi_u$ . Implementing the approach for a ZCS with state decision variables yields an affine transformation  $\Phi_x$ .

This procedure can be applied to multiple inputs and states. The corresponding vector functions for the affine transformations are given by

$$\Phi_u = \begin{bmatrix} \Phi_{u,1} \\ \vdots \\ \Phi_{u,n_u} \end{bmatrix} \quad \text{and} \quad \Phi_x = \begin{bmatrix} \Phi_{x,1} \\ \vdots \\ \Phi_{x,n_x} \end{bmatrix} \quad (38)$$

with  $\Phi_{u,i}$  and  $\Phi_{x,j}$  representing the transformation for the individual input and state, respectively. If no transformation is necessary, a direct mapping is implemented according to  $\Phi_{u,i} = u_i$  and  $\Phi_{x,i} = x_i$ . The vectors of transformations (38) are used to generate  $\Phi$  in (17) which is used for the convex depiction of the original constraints in (15b)–(15d).

### 2.3.2 Convexification of nonlinear equality constraints

Some nonlinearities in physical systems are characterised by a univariate function. Examples are nonlinear springs, saturation, dead-zones or the tire force shape curve of vehicles. Since nonlinear system equations yield nonlinear thus nonconvex equality constraints, this section presents how such nonlinear univariate terms can be convexified. The nonlinear univariate curve is depicted using piecewise affine parts. Hence, if the nonlinearity is not a piecewise linear function, SSC introduces approximation errors. The subsequent paragraphs illustrate the application of the SSC algorithm for convexifying piecewise linear nonlinearities.

**Theorem 2.6:** *The splitting constraints in (9) can be used to convexify equality constraints by transforming a univariate nonlinearity with two piecewise linear segments into an expression which is affine in multiple decision variables.*

**Proof:** Proof is provided using the piecewise linear curve depicted in Figure 5(a) as example, which represents a typical function for piecewise linear springs. Assuming the position  $x$  is a state of the system, it represents a decision variable which will be split at the transition point  $x = x_{tr} < 0$ . Thus, the auxiliary position variables  $x_{low} \in \mathcal{X}_{low} := [x, x_{tr}]$  and  $x_{up} \in \mathcal{X}_{up} := [x_{tr}, \bar{x}]$  are introduced. By including analogous constraints to (9a) and (11a) as well as additional objective terms in form of (11c), following relations are ensured using Lemma 2.1:

$$x_{low} = \begin{cases} x & \forall x \leq x_{tr}, \\ x_{tr} & \text{else} \end{cases}, \quad x_{up} = \begin{cases} x & \forall x \geq x_{tr}, \\ x_{tr} & \text{else} \end{cases}. \quad (39)$$

Hence, the piecewise linear function can be represented by the expression

$$\begin{aligned} \Phi_{pwl,2} &= c_{up}x_{up} + c_{low}(x_{low} - x_{tr}) \\ &= [c_{up} \quad c_{low}] \underbrace{\begin{bmatrix} x_{up} \\ x_{low} \end{bmatrix}}_{=:\mathbf{w}_{aux}} + (-c_{low}x_{tr}) \end{aligned} \quad (40)$$

which is affine in the decision variables  $\mathbf{w}_{aux}$ . The relations in (39) yield following two segments for the piecewise linear function (40):

$$\text{I} : x_{tr} \leq x \Rightarrow \begin{cases} x_{up} = x \\ x_{low} = x_{tr} \end{cases} \Rightarrow \Phi_{pwl,2} = c_{up}x. \quad (41a)$$

$$\begin{aligned} \text{II} : x < x_{tr} \Rightarrow \begin{cases} x_{up} = x_{tr} \\ x_{low} = x \end{cases} \\ \Rightarrow \Phi_{pwl,2} = c_{up}x_{tr} + c_{low}(x - x_{tr}). \end{aligned} \quad (41b)$$

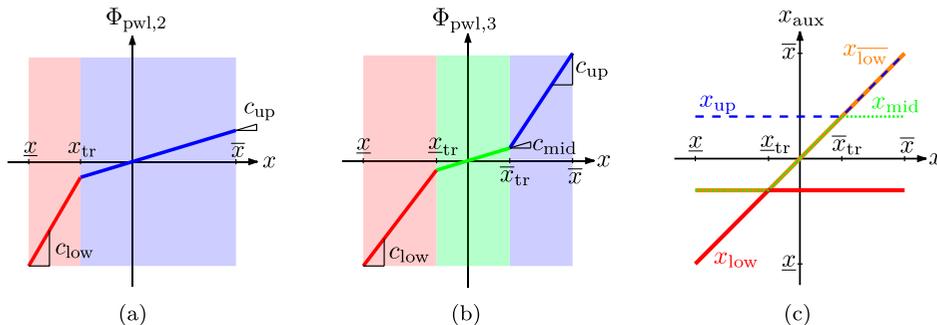
Thus, (41a) and (41b) depict the upper and lower segment of the nonlinearity, respectively. ■

Since (40) is an affine function in the decision variables, it can be inserted into the differential equations without preventing convexity. The subsequent theorem proves that this approach can be extended to univariate nonlinearities with more than two piecewise linear segments:

**Theorem 2.7:** *A repeated application of the space bisection approach using the splitting constraints in (9) can be used to convexify equality constraints by transforming a univariate nonlinearity with more than two piecewise linear segments into an expression which is affine in multiple decision variables.*

**Proof:** The general procedure for this extension is illustrated using the piecewise linear curve depicted in Figure 5(b). The position  $x$  is split at the transition points  $x = x_{tr} < 0$  and  $x = \bar{x}_{tr} > 0$ . Thus, the auxiliary position variables  $x_{low} \in \mathcal{X}_{low} := [x, x_{tr}]$ ,  $x_{low}^* \in \mathcal{X}_{low}^* := [x_{tr}, \bar{x}_{tr}]$ ,  $x_{mid} \in \mathcal{X}_{mid} := [\bar{x}_{tr}, \bar{x}]$  and  $x_{up} \in \mathcal{X}_{up} := [\bar{x}, \bar{x}]$  are employed. The space is repeatedly bisected following Lemma 2.1 to gain

$$x_{low} = \begin{cases} x & \forall x \leq x_{tr}, \\ x_{tr} & \text{else} \end{cases}, \quad x_{low}^* = \begin{cases} x & \forall x \geq x_{tr}, \\ x_{tr} & \text{else} \end{cases} \quad (42a)$$



**Figure 5.** Space splitting convexification of piecewise linear equalities. (a) Nonlinearity with two linear segments. (b) Nonlinearity with three linear segments. (c) Splitting of state variable  $x$  for three segments.

$$x_{\text{mid}} = \begin{cases} \underline{x}_{\text{low}} & \forall x_{\text{low}} \leq \bar{x}_{\text{tr}}, \\ \bar{x}_{\text{tr}} & \text{else} \end{cases}, x_{\text{up}} = \begin{cases} \underline{x}_{\text{low}} & \forall x_{\text{low}} \geq \bar{x}_{\text{tr}}, \\ \bar{x}_{\text{tr}} & \text{else} \end{cases} \quad (42b)$$

which is depicted in Figure 5. Thus, the position space is bisected into a region below and a region above  $x = \underline{x}_{\text{tr}}$  whereas the upper region is again bisected into a region below and above  $x = \bar{x}_{\text{tr}}$ . The piecewise linear function can then be represented by the expression

$$\Phi_{\text{pwl},3} = \begin{bmatrix} c_{\text{up}} & c_{\text{mid}} & c_{\text{low}} \end{bmatrix} \underbrace{\begin{bmatrix} x_{\text{up}} \\ x_{\text{mid}} \\ x_{\text{low}} \end{bmatrix}}_{=: \mathbf{w}_{\text{aux}}} - (c_{\text{up}}\bar{x}_{\text{tr}} + c_{\text{low}}\underline{x}_{\text{tr}}) \quad (43)$$

which is affine in the decision variables  $\mathbf{w}_{\text{aux}}$ . The relations in (42) yield following three segments for the piecewise linear function (43):

$$\text{I} : \bar{x}_{\text{tr}} < x \Rightarrow \begin{cases} x_{\text{up}} = x \\ x_{\text{mid}} = \bar{x}_{\text{tr}} \\ x_{\text{low}} = \underline{x}_{\text{tr}} \end{cases} \Rightarrow \Phi_{\text{pwl},3} = c_{\text{mid}}\bar{x}_{\text{tr}} + c_{\text{up}}(x - \bar{x}_{\text{tr}}). \quad (44a)$$

$$\text{II} : \underline{x}_{\text{tr}} \leq x \leq \bar{x}_{\text{tr}} \Rightarrow \begin{cases} x_{\text{up}} = \bar{x}_{\text{tr}} \\ x_{\text{mid}} = x \\ x_{\text{low}} = \underline{x}_{\text{tr}} \end{cases} \Rightarrow \Phi_{\text{pwl},3} = c_{\text{mid}}x. \quad (44b)$$

$$\text{III} : x < \underline{x}_{\text{tr}} \Rightarrow \begin{cases} x_{\text{up}} = \bar{x}_{\text{tr}} \\ x_{\text{mid}} = \underline{x}_{\text{tr}} \\ x_{\text{low}} = x \end{cases} \Rightarrow \Phi_{\text{pwl},3} = c_{\text{mid}}\underline{x}_{\text{tr}} + c_{\text{low}}(x - \underline{x}_{\text{tr}}). \quad (44c)$$

Hence, (44a), (44b) and (44c) depict the upper, mid and lower segment of the nonlinearity, respectively. ■

This repeated bisection procedure enables the generation of a characteristic curve with an arbitrary number of segments. However, an increasing number of segments also increases the number of auxiliary decision variables and additional constraints making the OP larger.

The approach presented in this section yields a scalar function  $\Phi_{\text{pwl}}$  which is affine in the decision variables. The procedure can be applied to multiple univariate, nonlinear terms yielding further scalar mapping functions. The affine vector-valued function in (16) is given by

$$\Phi_{\text{pwl}} = \begin{bmatrix} \Phi_{\text{pwl},1,1} + \Phi_{\text{pwl},1,2} + \dots \\ \vdots \\ \Phi_{\text{pwl},n_x,1} + \Phi_{\text{pwl},n_x,2} + \dots \end{bmatrix} \quad (45)$$

with  $\Phi_{\text{pwl},i,j}$  representing the individual affine mapping functions. For states that are not affected by the space splitting approach,  $\Phi_{\text{pwl},i,j} = 0$  holds. The transformation (45) is used to generate the convex collocation constraints (15b).

## 2.4 Remarks on runtime

The application of optimisation methods within a real-time capable controller generally requires upper bounds on the necessary computation time. The SSC approach runs through a loop which is comprised of a projection step and an optimisation step resulting in the following theorem regarding computational complexity:

**Theorem 2.8:** *The computational complexity of the SSC algorithm is polynomial.*

**Proof:** SSC solves a sequence of LP problems or convex QP problems, which are known to be solvable in polynomial time (Karmarkar, 1984; Kozlov et al., 1980). The projection step (12) of the SSC algorithm uses minimum and maximum functions which possess linear time complexity. This concludes a polynomial time complexity of the SSC algorithm:

$$T_{\text{SSC}}(n, q) = q \cdot (\mathcal{O}(n^\alpha) + c\mathcal{O}(n)) \quad \text{with } \alpha > 1. \quad (46)$$

Therein  $n$  can be interpreted as the number of operations required by the algorithm. The parameters in (46) depend on the utilised solver as well as the number of decision variables which is problem specific. Limiting the maximum number of superordinate iterations to  $q \leq q_{\text{max}}$  enables terminating the algorithm prematurely at suboptimal solutions, if necessary. ■

Theorem 2.8 provides valuable insight for the application in a real-time capable controller. Firstly, a worst-case bound on computation time can be experimentally identified for the specific OP and solver. Secondly, the computation time scales well with increasing problem size. Since the number of superordinate SSC iterations is rather low in practice, this results in a fast solution of optimal control problems. Finally, the influences on computation time are summarised below:

- (1) *System and discretisation.* The number of decision variables increases with rising number of inputs, states and collocation segments for both the original and the convexified OP. This increases computation time for NLP methods and the SSC algorithm. Due to the polynomial complexity, this increase in time will be generally less for SSC than for NLP.
- (2) *Nonlinearity.* The more complicated the nonlinearity, the more splitting segments are required to reduce approximation errors. However, the space splitting increases the problem size compared to the original problem, which influences the variable  $n$  in (46). This can put a practical bound on the number of space divisions via the SSC approach. Nevertheless, many physically motivated nonlinearities can be approximated sufficiently well using a moderate number of affine segments.
- (3) *Initial guess.* Being a local method, the solution computed by the SSC approach depends on the provided initial solution. Furthermore, this initial guess can influence the order of sign-corrections regarding the AVC-linearisations, which can impact the number of superordinate iterations and, therefore, the convergence rate. For MPC frameworks,

the optimal solution of the preceding time step often represents a good initial guess (Gros et al., 2020) promising low computation times.

- (4) *Penalty parameter update.* Too small or too large penalty parameters can slow down convergence (Nocedal & Wright, 2006, p. 511). The straightforward approach of gradually increasing the penalty parameter is used in this article. However, more sophisticated routines have been proposed in Mayne and Maratos (1979) and Maratos (1978).

## 2.5 Comparison with existing methods

In order to highlight the novelty of the proposed algorithm, this section compares the SSC approach with the most related existing methods.

From an algorithmic standpoint, linearising only the non-convex part of constraints combined with a relaxation and an increasing objective term penalising constraint violations corresponds to penalty CCP (Lipp & Boyd, 2016). CCP has the advantage over SQP and other linearisation techniques that it retains more problem information in each iterate: The information of the convex parts is kept and only the concave portions are linearised. Furthermore, CCP does not require to use line-search procedures or limit the progress in each iteration via trust-region methods. Inspired by Mangasarian (2015), the SSC approach avoids additional slack variables by considering the linearised constraints directly in the penalty objective. This avoids further increasing the size of the OP as in penalty CCP. Moreover, SSC employs an intermediate projection to enhance convergence. While CCP linearises the concave part of all non-convexities, SSC transforms the problem beforehand requiring only the linearisation of AVCs. This enables providing a feedback about the correctness of the selected linearisation points, which are subsequently adjusted. Thus, the linearisation error can vanish completely after the correct signs have been chosen. The binary nature of the linearisation facilitates a rapid convergence in a small number of superordinate iterations. Although the SSC method only computes solutions to the piecewise linear approximation of the original problem, the approximation can be designed to sufficient accuracy enabling small and known approximation errors. Especially the straightforward linearisation of ZCS-constraints yields very poor approximations, which can contain a large set of infeasible points. Thus, the vanishing AVC-linearisation errors result in an advantage of SSC regarding accuracy, also over further linearisation techniques like the ones presented in Mao et al. (2016, 2017). Generally, a scalable trade-off between accuracy and size of the OP, thus computation time, can be chosen. Opposed to Lipp and Boyd (2016), this article provides a convergence proof, even if only converging to the solution of the approximated problem.

While LC possesses the advantage of avoiding approximation errors, it is only applicable to OPs with annulus-like, non-convex sets resulting from excluding a convex subset from a convex set (Raković & Levine, 2018, p. 340). Opposed to that, SSC provides a method for the convexification of ZCS which cannot be considered via LC.

Similar to the LnP approach presented in Mao et al. (2017), the SSC method is an iterative linearisation algorithm using

intermediate projection steps. The updating routine for SSC is of linear computational complexity with few and simple mathematical operations. The projection step of the LnP approach generally requires the solution of a convex programming problem, which is of polynomial complexity at best. Thus, the projection step used by the SSC approach is computationally less costly, which is beneficial for splitting methods (Ferreau et al., 2017). Additionally, the binary nature of the AVC-linearisation provides the advantage that generally only few superordinate iterations are required, resulting in short computation times. Moreover, SSC is more robust regarding the initialisation since the LnP approach requires an additional lifting procedure to cope with arbitrary initialisations, which costs computation time (Mao et al., 2017). Furthermore, the LnP method requires the right-hand side of the system equations to be convex and the inequality constraints to be concave. However, nonconvex collision avoidance constraints can be easily considered via the LnP approach, whereas the SSC procedure requires further measures.

The ADMM is a splitting procedure utilising the augmented Lagrangian method instead of the exact penalty function method, which is employed by the SSC algorithm. Due to the alternating approach, the ADMM requires the solution of two QCQP problems in each iteration. The SSC approach only requires the solution of a single convex QP problem in each iteration, which is beneficial for computation time.

The spatial BnB method also uses space decomposition for the convexification of the optimisation problem. However, the SSC approach solves a piecewise linear approximation of the original problem, which is iteratively adjusted, on the full space domain of the decision variables. Opposed to that, the BnB method solves multiple OPs which represent convexly relaxed versions of the original problem on separate subdomains of the decision variables. BnB techniques determine upper bounds by evaluating the objective function at a feasible point. Since the problem is generally nonconvex, finding a feasible point can be difficult. This is often tackled by solving the nonconvex subproblem locally, which is computationally expensive (Liberti & Maculan, 2006, p. 253). Thus, the convergence rate of the SSC approach is most likely to be superior if sufficiently good initial guesses are provided. However, this depends on the nonlinearities of the OP, which influence the number of decision variables for SSC but also for BnB approaches which decompose factorable functions (Tawarmalani & Sahinidis, 2002, p. 125). Moreover, SSC only guarantees to find local optima of an approximation of the OP and is only applicable to certain classes of OPs. GO techniques cover a wide range of applicability and provide the global optimum.

## 3. Applications

In this section, the proposed SSC algorithm is evaluated using a hanging SMO example with piecewise linear spring and semi-active damping as illustrated in Figure 6. Semi-active dampers can only generate a force in the opposite moving direction and cannot impose a force at steady-state complying with the passivity constraint (Savaresi et al., 2010, p. 16). Hence, the admissible set for semi-active dampers always contains a subset in the first quadrant and a subset in the third quadrant which are connected

in a single point, namely, the origin. A performance-oriented OP is formulated and solved using NLP and SSC, respectively. The results of both algorithms are compared regarding accuracy and computation time. Both problems are posed using the modelling language *JuMP* (Dunning et al., 2017) for mathematical OPs. For a fair comparison, the necessary derivatives are computed beforehand via automatic differentiation. Furthermore, the tolerances for constraint feasibility and objective improvement are set to  $10^{-6}$  for both solvers. The utilised parameters are listed with description in Table 1.

The NLP problem is solved using the optimiser *IPOPT* (Wächter & Biegler, 2006), which employs an interior-point method: Using the concept of barrier functions, a sequence of relaxed problems is solved for decreasing values of barrier parameters. The barrier parameter relaxes the inequality constraints and hence represents the distance from the current iterate to the border of the admissible set. Thus, interior-point methods use a central path within the admissible set as opposed to SQP-methods which wander along the border. The resulting nonlinear equation system is solved via a damped Newton method. Furthermore, a filter line-search method, heuristics and further correction measures are employed to enhance convergence and robustness.

The novel SSC method transforms the OP into convex QP subproblems which are solved iteratively. In order to capitalise on this problem structure, the elaborate optimiser *Gurobi* (Gurobi Optimization, 2020) is selected.

**Table 1.** Parameters for SMO application.

Symbol	Value	Unit	Description
$g$	9.81	m/s <sup>2</sup>	gravitational acceleration
$m$	5.00	kg	mass
$c_{up}$	10.00	N/m	overload spring stiffness for rebound
$c_{mid}$	3.00	N/m	main spring stiffness
$c_{low}$	5.00	N/m	overload spring stiffness for compression
$\bar{x}_{tr}$	5.00	m	transition point for rebound overload
$\underline{x}_{tr}$	-5.00	m	transition point for compression overload
$x_{ss,2}$	16.35	m	steady-state position for 2-segment spring
$x_{ss,3}$	8.405	m	steady-state position for 3-segment spring
$\bar{x}$	100.00	m	upper position bound
$\underline{x}$	-100.00	m	lower position bound
$\bar{d}$	20.00	Ns/m	upper bound on damper coefficient
$\underline{d}$	0.50	Ns/m	lower bound on damper coefficient
$\bar{v}$	100.00	m/s	upper speed bound
$\underline{v}$	-100.00	m/s	lower speed bound
$\bar{F}_d$	400.00	N	upper bound on damper force
$\underline{F}_d$	-400.00	N	lower bound on damper force
$\varepsilon_g$	$10^{-6}$	-	violation tolerance for SSC penalty term
$\underline{\tau}/\bar{\tau}$	$1.00/10^4$	-	initial/final value for penalty parameter

### 3.1 Optimisation problems

The equations of motion for the state vector  $\mathbf{x} = [x \ v]^T$  result in

$$\dot{\mathbf{x}} = \mathbf{f} = \begin{bmatrix} v \\ \frac{1}{m} (mg - F_d - F_c) \end{bmatrix} \quad (47)$$

with  $x$ ,  $v$ ,  $F_d$  and  $F_c$  denoting the deflection position, deflection speed, damper force and spring force, respectively. The expression of these forces differs for the NLP method and the SSC approach. Starting from a specified initial state  $\mathbf{x}_{iv} = [x_{iv} \ v_{iv}]^T$  with damper force  $F_{d,iv} = \underline{d}v_{iv}$ , the goal is to minimise the deviation from the steady-state position  $x_{ss}$  while considering the limitations of the system. Applying Simpson quadrature (Betts, 2010; Kelly, 2017), the main objective penalises the steady-state position deviation  $e_{ss,j} := x_j - x_{ss}$  according to

$$J_{ss} := \sum_{i=0}^{n_{seg}-1} \frac{\Delta_i}{6} \left( e_{ss,i}^2 + 4e_{ss,i+\frac{1}{2}}^2 + e_{ss,i+1}^2 \right). \quad (48)$$

Considering the values listed in Table 1, the steady-state position can be computed using a root-finding approach like Newton's method: At steady-state position, the gravitational force must equal the spring force. For the considered nonlinear springs with two piecewise linear segments the value  $x_{ss} = x_{ss,2}$  is chosen and  $x_{ss} = x_{ss,3}$  if the spring with three segments is used. A uniform discretisation of the time grid  $t \in [t_0, t_f] = [0, 10]$  is employed yielding a constant segment width  $\Delta_i = \Delta = \frac{t_f - t_0}{n_{seg}}$ . Both, the NLP approach and the SSC method, are initialised with zero vectors  $\mathbf{U}_0^* = \mathbf{0}$  and  $\mathbf{X}_0^* = \mathbf{0}$ . From

$$\begin{aligned} f_{ss,i}(x) &:= \frac{\Delta_i}{6} (x_i - x_{ss})^2 \Rightarrow f'_{ss,i}(x) = 2 \frac{\Delta_i}{6} (x_i - x_{ss}) \\ &\Rightarrow f''_{ss,i}(x) = 2 \frac{\Delta_i}{6} > 0 \end{aligned} \quad (49)$$

follows that the Hessian of (48) is a diagonal matrix with only positive and zero diagonal entries and therefore positive semi-definite. Hence, the main objective (48) is a convex function.

#### 3.1.1 Nonlinear nonconvex optimisation problem

For the solution of the OP via NLP, the decision variables are chosen as

$$\mathbf{u} = [u_0 \ u_{0.5} \ \dots \ u_{n_{seg}}] \in \mathbb{R}^{1 \times n_{pts}} \text{ with } u_k = d_k \quad (50a)$$

$$\mathbf{X} = [\mathbf{x}_0 \ \mathbf{x}_{0.5} \ \dots \ \mathbf{x}_{n_{seg}}] \in \mathbb{R}^{2 \times n_{pts}} \text{ with } \mathbf{x}_k := \begin{bmatrix} x_k \\ v_k \end{bmatrix} \quad (50b)$$

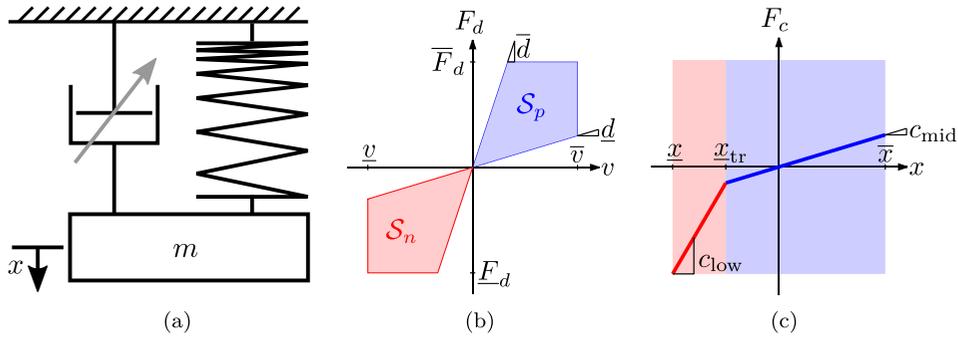
whereas the input represents the variable damping coefficient  $d_k$  of the semi-active actuator. With  $i \in \mathcal{I}$  and  $k \in \mathcal{K}$ , the original OP is given by

$$\min_{\mathbf{u}, \mathbf{X}} J_{ss} \quad (51a)$$

$$\text{s.t. } \mathbf{c}_{coll,i} \stackrel{(5)}{=} \mathbf{0} \text{ with } \mathbf{f}_k = \begin{bmatrix} v_k \\ \frac{1}{m} (mg - F_{d,k}^{NLP} - F_{c,2,k}^{NLP}) \end{bmatrix}, \quad (51b)$$

$$F_{d,k}^{NLP} = u_k v_k \text{ and}$$

$$F_{c,2,k}^{NLP} = \min_{\varepsilon} (c_{mid} x_k, c_{low} x_k - \underline{x}_{tr} (c_{low} - c_{mid})),$$



**Figure 6.** Model characteristics for hanging SMO with semi-active damper and nonlinear spring. (a) Hanging SMO model. (b) ZCS for semi-active damper force. (c) Piecewise linear spring force characteristic.

$$u_k \in \mathcal{D} := [\underline{d}, \bar{d}], \quad x_k \in \mathcal{X} := [\underline{x}, \bar{x}] \quad \text{and} \quad (51c)$$

$$v_k \in \mathcal{V} := [\underline{v}, \bar{v}],$$

$$\underline{F}_d \leq u_k v_k \leq \bar{F}_d, \quad (51d)$$

$$u_0 = \underline{d}, \quad \mathbf{x}_0 = \mathbf{x}_{iv}. \quad (51e)$$

The nonconvex NLP problem (51) is comprised of following parts: objective function (51a), collocation constraints (51b), bounds for the decision variables (51c), damper force bounds (51d) and initial value condition (51e). The piecewise linear spring force is approximated in (51b) via the smoothed minimum-function

$$\min_\varepsilon(x, y) := \frac{1}{2} \left( (x + y) - \sqrt{(x - y)^2 + \varepsilon} \right) \quad (52)$$

using  $\varepsilon = 10^{-16}$  to guarantee that the problem is twice continuously differentiable (Sedlacek et al., 2020b). Problem (51) is nonconvex due to the right-hand side of the differential equation system in the collocation constraints (51b) and due to the damper force bounds (51d).

### 3.1.2 Convex QP-problem for iterative solution

Following the SSC approach, auxiliary optimisation variables are introduced, which results in the subsequent decision variables:

$$\mathbf{U} = [\mathbf{u}_0 \quad \mathbf{u}_{0.5} \quad \dots \quad \mathbf{u}_{n_{\text{seg}}}] \in \mathbb{R}^{2 \times n_{\text{pts}}} \quad \text{with} \quad \mathbf{u}_k := \begin{bmatrix} u_{\text{pos},k} \\ u_{\text{neg},k} \end{bmatrix} = \begin{bmatrix} F_{d,\text{pos},k} \\ F_{d,\text{neg},k} \end{bmatrix} \quad (53a)$$

$$\mathbf{X} = [\mathbf{x}_0 \quad \mathbf{x}_{0.5} \quad \dots \quad \mathbf{x}_{n_{\text{seg}}}] \in \mathbb{R}^{2 \times n_{\text{pts}}} \quad \text{with} \quad \mathbf{x}_k := \begin{bmatrix} x_k \\ v_k \end{bmatrix} \quad (53b)$$

$$\mathbf{X}_{\text{aux}} = [\mathbf{x}_{\text{aux},0} \quad \mathbf{x}_{\text{aux},0.5} \quad \dots \quad \mathbf{x}_{\text{aux},n_{\text{seg}}}] \in \mathbb{R}^{2 \times n_{\text{pts}}} \quad \text{with} \quad \mathbf{x}_{\text{aux},k} := \begin{bmatrix} x_{\text{mid},k} \\ x_{\text{low},k} \end{bmatrix} \quad (53c)$$

$$\mathbf{V}_{\text{aux}} = [\mathbf{v}_{\text{aux},0} \quad \mathbf{v}_{\text{aux},0.5} \quad \dots \quad \mathbf{v}_{\text{aux},n_{\text{seg}}}] \in \mathbb{R}^{2 \times n_{\text{pts}}} \quad \text{with} \quad \mathbf{v}_{\text{aux},k} := \begin{bmatrix} v_{\text{pos},k} \\ v_{\text{neg},k} \end{bmatrix}. \quad (53d)$$

Opposed to the inputs (50a) of the NLP problem (51a), the inputs in (53a) represent the positive and negative part of the damper force.

As illustrated in Algorithm 1, a projection routine is executed in each iteration. Based on the preceding solution marked by values  $(\cdot)^*$ , this routine computes the projected initial solution  $(\check{\cdot})$  for  $k \in \mathcal{K}$  via

$$\check{\mathbf{u}}_k = \begin{bmatrix} \check{u}_{\text{pos},k} \\ \check{u}_{\text{neg},k} \end{bmatrix} = \begin{bmatrix} \max(u_{\text{neg},k}^* + u_{\text{pos},k}^*, 0) \\ \min(u_{\text{neg},k}^* + u_{\text{pos},k}^*, 0) \end{bmatrix}, \quad \check{\mathbf{x}}_k = \mathbf{x}_k^* \quad (54a)$$

$$\underbrace{\begin{bmatrix} \check{x}_{\text{mid},k} \\ \check{x}_{\text{low},k} \end{bmatrix}}_{=\check{\mathbf{x}}_{\text{aux},k}} = \begin{bmatrix} \max(x_k^*, x_{\text{tr}}) \\ \min(x_k^*, x_{\text{tr}}) \end{bmatrix}, \quad \underbrace{\begin{bmatrix} \check{v}_{\text{pos},k} \\ \check{v}_{\text{neg},k} \end{bmatrix}}_{=\check{\mathbf{v}}_{\text{aux},k}} = \begin{bmatrix} \max(v_k^*, 0) \\ \min(v_k^*, 0) \end{bmatrix} \quad (54b)$$

and the correct signs according to

$$\sigma_{x,k} = \text{sign}(x_k^* - x_{\text{tr}}), \quad \sigma_{v,k} = \text{sign}(v_k^*). \quad (54c)$$

For a concise notation in the OP, the decision variables are lumped together in  $\mathbf{P} := \{\mathbf{U}, \mathbf{X}, \mathbf{X}_{\text{aux}}, \mathbf{V}_{\text{aux}}\}$ . With  $i \in \mathcal{I}$  and  $k \in \mathcal{K}$ , the convex QP problem, which is iteratively solved for updated values of  $\tau$ ,  $\sigma_{x,k}$ ,  $\sigma_{v,k}$  and corresponding initial guesses, is given by

$$\min_{\mathbf{P}} \quad J_{\text{ss}} + \tau \sum_{k \in \mathcal{K}} l_{\text{abs},x,k} + l_{\text{abs},v,k} \quad \text{with} \quad (55a)$$

$$\text{s.t.} \quad \begin{aligned} l_{\text{abs},x,k} &:= (x_{\text{mid},k} - x_{\text{low},k}) - \sigma_{x,k}(x_k - x_{\text{tr}}) \\ l_{\text{abs},v,k} &:= (v_{\text{pos},k} - v_{\text{neg},k}) - \sigma_{v,k}v_k, \end{aligned} \quad (55b)$$

$$\mathbf{c}_{\text{coll},i} \stackrel{(5)}{=} \mathbf{0} \quad \text{with} \quad \mathbf{f}_k = \begin{bmatrix} v_k \\ \frac{1}{m} (mg - F_{d,k}^{\text{SSC}} - F_{c,2,k}^{\text{SSC}}) \end{bmatrix}, \quad (55c)$$

$$F_{d,k}^{\text{SSC}} = u_{\text{neg},k} + u_{\text{pos},k} \quad \text{and}$$

$$F_{c,2,k}^{\text{SSC}} = c_{\text{mid}}x_{\text{mid},k} + c_{\text{low}}(x_{\text{low},k} - x_{\text{tr}}),$$

$$x_k \in \mathcal{X} := [\underline{x}, \bar{x}], \quad v_k \in \mathcal{V} := [\underline{v}, \bar{v}],$$

$$u_{\text{neg},k} \in \mathcal{U}_{\text{neg}} := [\underline{F}_d, 0], \quad u_{\text{pos},k} \in \mathcal{U}_{\text{pos}} := [0, \bar{F}_d], \quad (55d)$$

$$v_{\text{neg},k} \in \mathcal{V}_{\text{neg}} := [\underline{v}, 0], \quad v_{\text{pos},k} \in \mathcal{V}_{\text{pos}} := [0, \bar{v}],$$

$$x_{\text{low},k} \in \mathcal{X}_{\text{low}} := [\underline{x}, x_{\text{tr}}], \quad x_{\text{mid},k} \in \mathcal{X}_{\text{mid}} := [x_{\text{tr}}, \bar{x}],$$

$$\hat{\mathbf{h}}_{\text{zcs},p,k} = \begin{bmatrix} -\bar{d} & 1 \\ \underline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\text{pos},k} \\ u_{\text{pos},k} \end{bmatrix} \leq \mathbf{0}, \quad (55e)$$

$$\hat{\mathbf{h}}_{\text{zcs},n,k} = \begin{bmatrix} -\bar{d} & 1 \\ \underline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\text{neg},k} \\ u_{\text{neg},k} \end{bmatrix} \leq \mathbf{0},$$

$$g_{\text{aff},x,k} = x_{\text{mid},k} + x_{\text{low},k} - (x_k + \underline{x}_{\text{tr}}) = 0, \quad (55f)$$

$$g_{\text{aff},v,k} = v_{\text{pos},k} + v_{\text{neg},k} - v_k = 0,$$

$$\mathbf{h}_{\text{abs},x,k} = \begin{bmatrix} l_{\text{abs},x,k} |_{\sigma_{x,k}=+1} \\ l_{\text{abs},x,k} |_{\sigma_{x,k}=-1} \end{bmatrix} \geq \mathbf{0}, \quad (55g)$$

$$\mathbf{h}_{\text{abs},v,k} = \begin{bmatrix} l_{\text{abs},v,k} |_{\sigma_{v,k}=+1} \\ l_{\text{abs},v,k} |_{\sigma_{v,k}=-1} \end{bmatrix} \geq \mathbf{0},$$

$$\mathbf{u}_0 = \begin{bmatrix} \max(dv_{\text{iv}}, 0) \\ \min(dv_{\text{iv}}, 0) \end{bmatrix}, \quad \mathbf{x}_0 = \mathbf{x}_{\text{iv}}. \quad (55h)$$

The QP problem (55) is comprised of following parts: augmented objective function (55a), collocation constraints (55c), bounds for the decision variables (55d), ZCS-constraints (55e), the additional space splitting constraints (55f)–(55g) and initial value condition (55h). Due to the splitting approach, the right-hand side of the differential equation system in (55c) is an affine function in the decision variables. Thus, all constraints are affine in the decision variables and therefore convex. The additional objectives with (55b) are affine in the decision variables representing convex functions. Hence, (55) represents a convex QP problem. Using (55b), the termination criterium in line 9 of Algorithm 1 is computed via

$$\bar{l}_{\text{abs}} = \left\| \begin{bmatrix} l_{\text{abs},x,0} \\ l_{\text{abs},v,0} \end{bmatrix}, \dots, \begin{bmatrix} l_{\text{abs},x,n_{\text{seg}}} \\ l_{\text{abs},v,n_{\text{seg}}} \end{bmatrix} \right\|_{\max}. \quad (56)$$

As illustrated in line 6 of Algorithm 1, the penalty weight  $\tau$  is linearly increased with a maximum iteration number  $q_{\text{max}} = 6$  using the lower and upper bound  $\underline{\tau}$  and  $\bar{\tau}$ , respectively. Both OPs are now fully defined. The results generated by solving NLP problem (51) and by applying the SSC algorithm with QP problem (55) are compared in the following section.

### 3.2 Results

In order to inspect the performance of the proposed SSC algorithm, 100 OPs are analysed in this section using a desktop computer with Intel Core™ i7-9850H CPU to determine the solutions. The OPs vary in size and the initial value condition. The number of collocation segments is gradually increased with  $n_{\text{seg}} \in \{10, 50, 100, 250, 500\}$ . For each problem size, the OPs (51) and (55) are solved using the 10 random, admissible initial values  $\mathbf{x}_{\text{iv}}$  listed in Table 2. This procedure is also executed for a simplified QP problem (55) with a fully linear spring of stiffness  $c_{\text{mid}}$ . This simplifies the right-hand side of the differential equation system in (55c) and eliminates the need for the auxiliary variables  $\mathbf{X}_{\text{aux}}$  in (53c), the corresponding additional objectives in (55b) as well as the corresponding constraints in (55f) and (55g). The optimisation results are displayed in Table 3.

Firstly, the OPs with initial value  $\mathbf{x}_{\text{iv}} = \mathbf{x}_{\text{iv},1}$  and  $n_{\text{seg}} = 250$  segments are studied in more detail. Figure 7(a) depicts the

piecewise linear characteristic curve of the spring force and Figure 7(b) the admissible set for the damper force. The resulting trajectories of the states and damper force are illustrated in Figure 7(c). The mean absolute deviance between the damper force trajectory of the NLP solution and the SSC solution is defined as

$$\bar{e}_{F_d} := \frac{1}{n_{\text{pts}}} \sum_{k \in \mathcal{K}} |F_{d,k}^{\text{NLP}} - F_{d,k}^{\text{SSC}}| \quad (57)$$

for the  $n_{\text{pts}}$  collocation points. With  $\bar{e}_{F_d} = 1.85\text{N}$ , the control trajectories differ only marginally considering the damper force domain  $F_d \in [-400\text{N}, 400\text{N}]$ . The mean absolute deviance between the position and the steady-state position

$$\bar{e}_{\text{ss}} := \frac{1}{n_{\text{pts}}} \sum_{k \in \mathcal{K}} |x_k - x_{\text{ss},k}| \quad (58)$$

is  $\bar{e}_{\text{ss}}^{\text{SSC}} = 7.12\text{m}$  and  $\bar{e}_{\text{ss}}^{\text{NLP}} = 7.18\text{m}$  for the SSC solution and NLP solution, respectively. However, the value for the main cost function (48) is  $J_{\text{ss}}^{\text{SSC}} = 2550.73\text{m}^2\text{s}$  and  $J_{\text{ss}}^{\text{NLP}} = 2549.86\text{m}^2\text{s}$  for the SSC solution and NLP solution, respectively. Thus, the SSC objective value is higher but the mean deviance in the steady-state position error is lower. The reason for this is that the cost function is quadratic in the errors which penalises large errors unevenly more than small errors. As illustrated in the top subplot of Figure 7, the position of the NLP solution reaches the steady-state position faster however with an overshoot which slowly reduces. Thus, the larger errors at the beginning are reduced more quickly resulting in a lower cost function value. However, the mean position error, which is a better metric for the actual goal of minimising the deviance from the steady-state position, is worse than for the SSC solution. Although an objective penalising the absolute value of the errors would be more adequate, it is not used since it would prevent continuous differentiability of the objective function. Nevertheless, both algorithms reach the steady-state position  $x_{\text{ss}}$  in about 3 s resulting in similar state trajectories. The trajectories of the auxiliary variables verify correct switching at  $x = \underline{x}_{\text{tr}} = -5\text{m}$  and  $v = v_{\text{tr}} = 0\frac{\text{m}}{\text{s}}$  for the position and velocity, respectively. Furthermore, Figure 7(a,b) shows that the spring forces lie on the characteristic line and the computed damper forces lie within the admissible set proving the compliance of the prescribed constraints. The bang-bang-like control strategy confirms correctness since it is expected due to the main objective (48), which demands to reduce the errors as fast as possible. With  $q = 4$  superordinate iterations, the SSC method required a total optimisation time of  $t_{\Sigma}^{\text{SSC}} = 0.329\text{s}$  which is 47.71 % of the computation time of  $t_{\Sigma}^{\text{NLP}} = 0.690\text{s}$  required by the NLP solver. The cumulative optimisation time  $t_{\text{opt},\Sigma}$ , which represents the time spent in the QP solver in line 8 of the SSC algorithm for all passed loops, is  $t_{\text{opt},\Sigma}^{\text{SSC}} = 0.262\text{s}$  representing 37.97 % of the NLP

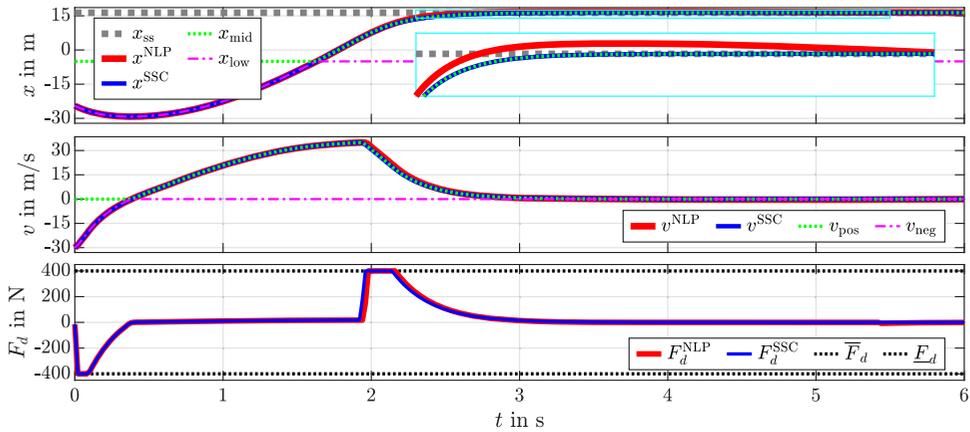
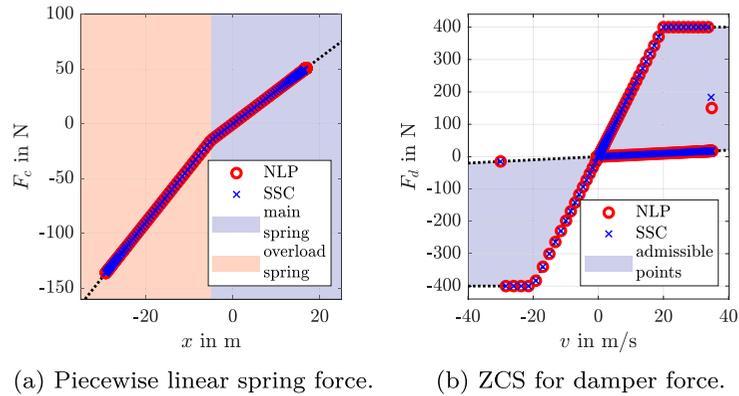
**Table 2.** Initial values of robustness analysis.

	$x_{\text{iv},1}$	$x_{\text{iv},2}$	$x_{\text{iv},3}$	$x_{\text{iv},4}$	$x_{\text{iv},5}$	$x_{\text{iv},6}$	$x_{\text{iv},7}$	$x_{\text{iv},8}$	$x_{\text{iv},9}$	$x_{\text{iv},10}$
$x$	−24.5	−15.1	−51.4	27.9	59.4	12.3	56.1	15.4	−73.7	−56.9
$v$	−30.0	−44.9	−15.4	39.3	18.7	64.3	−15.5	−32.7	16.8	73.1

**Table 3.** Optimisation results of robustness analysis.

$n_{\text{seg}}[-]$	Spring		Method		Results						
	LIN	NL	NLP	SSC	$\bar{J}_{ss}$ [m <sup>2</sup> s]	$\bar{e}_{ss}$ [m]	$\bar{e}_{F_d}$ [N]	$\bar{q}[-]$	$t_{\text{opt},\Sigma}$ [s]	$t_{\Sigma}$ [s]	
10	✓		✓		815.26	7.64	–	1.00	0.074	0.074	
10	✓			✓	818.92	7.50	3.51	2.00	0.005 (6.45%)	0.021 (27.77%)	
10		✓	✓		597.24	7.11	–	1.00	0.075	0.075	
10		✓		✓	601.84	6.96	2.72	2.40	0.008 (10.90%)	0.025 (33.67%)	
50	✓		✓		2135.09	6.67	–	1.00	0.125	0.125	
50	✓			✓	2139.30	6.55	2.04	2.00	0.016 (12.87%)	0.036 (28.51%)	
50		✓	✓		1842.48	6.16	–	1.00	0.127	0.127	
50		✓		✓	1849.97	6.03	2.28	2.60	0.036 (28.46%)	0.059 (46.34%)	
100	✓		✓		2333.84	6.52	–	1.00	0.229	0.229	
100	✓			✓	2338.26	6.41	1.94	2.00	0.026 (11.18%)	0.050 (21.77%)	
100		✓	✓		2043.91	6.02	–	1.00	0.252	0.252	
100		✓		✓	2051.33	5.89	2.20	2.70	0.064 (25.44%)	0.095 (33.53%)	
250	✓		✓		2454.05	6.43	–	1.00	0.587	0.587	
250	✓			✓	2458.40	6.32	1.94	2.00	0.068 (11.64%)	0.105 (18.20%)	
250		✓	✓		2167.57	5.93	–	1.00	0.695	0.695	
250		✓		✓	2174.81	5.80	2.22	2.70	0.183 (26.30%)	0.235 (33.83%)	
500	✓		✓		2494.96	6.40	–	1.00	1.080	1.080	
500	✓			✓	2499.36	6.29	1.97	2.10	0.173 (16.03%)	0.235 (21.72%)	
500		✓	✓		2209.03	5.91	–	1.00	1.373	1.373	
500		✓		✓	2216.23	5.77	2.22	2.80	0.460 (33.50%)	0.555 (40.39%)	

$n_{\text{seg}}$ : # of collocation segments; LIN/NL: linear/nonlinear spring; NLP: IPOPT solving NLP problem (51); SSC: SSC Algorithm 1 with QP problem (55);  $\bar{J}_{ss}, \bar{e}_{ss}, \bar{e}_{F_d}$ : main objective and error metrics (59) (average for 10 initial value conditions);  $\bar{q}$ : average # of superordinate iterations;  $t_{\text{opt},\Sigma}$ : cumulative time spent in NLP/QP solver;  $t_{\Sigma}$ : total computation time.



(c) State and damper force trajectories.

**Figure 7.** Solution of OPs with initial value  $x_{iv,1}$  and  $n_{\text{seg}} = 250$  collocation segments. (a) Piecewise linear spring force. (b) ZCS for damper force. (c) State and damper force trajectories.

time. Since the SSC algorithm converged in  $q < q_{\text{max}} = 6$  iterations, the largest relaxation value of the AVCs is smaller than  $\varepsilon_g$  ensuring conformity with the original problem.

Subsequently, the remaining results of Table 3 are analysed to identify tendencies. As stated in Section 2.2.1, the additional

objective terms (55b) represent a relaxation of the constraints comparable with slack variables. Besides serving as feedback for determining convergence, this relaxation provides robustness regarding the initialisation. This is reflected in the results by a successful convergence of the SSC algorithm in 100% of

the test cases, even though a rather poor initial guess was used. The accuracy of the computed solutions is evaluated using the mean values of the main cost function (48) and of the averaged errors (57) and (58) defined as

$$\bar{J}_{ss} := \sum_{i=1}^{10} J_{ss,i}, \quad \bar{e}_{F_d} := \sum_{i=1}^{10} \bar{e}_{F_d,i} \quad \text{and} \quad \bar{e}_{ss} := \sum_{i=1}^{10} \bar{e}_{ss,i}. \quad (59)$$

These values represent the respective mean value over all 10 OPs, due to 10 different initial values, for one specified number of collocation segments  $n_{seg}$ . With  $\bar{e}_{F_d} \leq 3.51$  N being small compared to the domain of  $F_d$ , the control trajectories of the SSC approach are close to the trajectories computed by the NLP solver. The small deviation of the mean values in (59) between the SSC solution and the NLP solution indicates that both algorithms converged to a similar solution. Considering the mean value  $\bar{J}_{ss}$  of the main objective function, the NLP solver yields lower objective values than the SSC algorithm. As previously discussed, the mean deviance  $\bar{e}_{ss}$  from the steady-state position is smaller when using the SSC algorithm.

The SSC approach greatly reduces the overall computation time  $t_{\Sigma}^{SSC}$  and requires up to only 18.20% of the computation time needed by the NLP solver. The time advantage of the SSC algorithm over the NLP optimiser will be even bigger for asymmetric zonally convex damper sets. Such asymmetric sets require additional discontinuities in the NLP problem but can be easily implemented within the SSC procedure by using differing parameters for the individual convex subsets. As mentioned in Section 2.4, various aspects influence the runtime of the SSC algorithm which is reflected in the results: With rising number of collocation segments, the mean solution time increases and the computation time of SSC increases less than the NLP solution time. Another mentioned effect on runtime is the quality of the initial guess. The provided guess with zero vectors is of varying quality for the individual initial value conditions. A better initial guess in terms of the correct signs of the state trajectories reduces the number of superordinate iterations and thus the overall computation time. Hence, the sensitivity in regards to the initial guess is elaborated in the following section.

The pure optimisation time  $t_{opt,\Sigma}^{SSC}$  reduces up to 6.45% of the corresponding NLP time and is generally significantly shorter than its overall computation time  $t_{\Sigma}^{SSC}$ . Since the projection step of the SSC algorithm employs only simple computations using minimum and maximum functions with linear time complexity, the time spent outside of the QP solver, which solves convex QP-problems in polynomial time, seems quite long. Unfortunately, substantial time losses occur at building the updated optimisation model via *JuMP* and the *Gurobi*-wrapper. For future implementations, it is advisable to implement the SSC algorithm via C-code to circumvent this problem and minimise the overall computation time.

### 3.3 Influence of initial guess on results

In this section, the OPs (51) and (55) are modified to consider the nonlinear spring characteristic with three piecewise linear segments depicted in Figure 5. The NLP problem (51) is adjusted by replacing the spring force  $F_{c,2,k}^{NLP}$  in the differential

equation system considered in constraint (51b) by

$$F_{c,3,k}^{NLP} = \max_{\varepsilon} [\min_{\varepsilon} (c_{mid}x_k, c_{low}x_k - \underline{x}_{tr}(c_{low} - c_{mid})), c_{up}x_k + \bar{x}_{tr}(c_{mid} - c_{up})] \quad (60)$$

using the smooth maximum function

$$\max_{\varepsilon}(x, y) := \frac{1}{2} \left( (x + y) + \sqrt{(x - y)^2 + \varepsilon} \right). \quad (61)$$

The adaptation of the QP problem (55) requires slightly more changes. The auxiliary decision variables for the position (53c) are augmented resulting in

$$\mathbf{x}_{aux,k} := [x_{low,k}^{\bar{}} \quad x_{low,k} \quad x_{up,k} \quad x_{mid,k}]^T. \quad (62)$$

Thus, the position space is bisected into a region below and a region above  $x = \underline{x}_{tr}$ , whereas the upper region is again bisected into a region below and above  $x = \bar{x}_{tr}$ . Compared to (54), the update procedure remains unchanged for the decision variables of the inputs, states and auxiliary velocities:

$$\begin{aligned} \begin{bmatrix} \check{u}_{pos,k} \\ \check{u}_{neg,k} \end{bmatrix} &= \begin{bmatrix} \max(u_{neg,k}^* + u_{pos,k}^*, 0) \\ \min(u_{neg,k}^* + u_{pos,k}^*, 0) \end{bmatrix}, \\ \hat{\mathbf{x}}_k = \mathbf{x}_k^*, \quad \begin{bmatrix} \check{v}_{pos,k} \\ \check{v}_{neg,k} \end{bmatrix} &= \begin{bmatrix} \max(v_k^*, 0) \\ \min(v_k^*, 0) \end{bmatrix}. \end{aligned} \quad (63a)$$

However, since the position space is partitioned into three segments, the updating routine changes for the auxiliary position states according to

$$\begin{aligned} \begin{bmatrix} \check{x}_{low,k}^{\bar{}} \\ \check{x}_{low,k} \end{bmatrix} &= \begin{bmatrix} \max(x_k^*, \underline{x}_{tr}) \\ \min(x_k^*, \underline{x}_{tr}) \end{bmatrix}, \\ \begin{bmatrix} \check{x}_{up,k} \\ \check{x}_{mid,k} \end{bmatrix} &= \begin{bmatrix} \max(\check{x}_{low,k}^{\bar{}}, \bar{x}_{tr}) \\ \min(\check{x}_{low,k}^{\bar{}}, \bar{x}_{tr}) \end{bmatrix}. \end{aligned} \quad (63b)$$

Furthermore, the correct signs are now given by

$$\begin{aligned} \sigma_{x_{mu},k} &= \text{sign}(\check{x}_{low,k}^{\bar{}} - \bar{x}_{tr}), \quad \sigma_{x_{lm},k} = \text{sign}(x_k^* - \underline{x}_{tr}), \\ \sigma_{v,k} &= \text{sign}(v_k^*). \end{aligned} \quad (63c)$$

With  $i \in \mathcal{I}$  and  $k \in \mathcal{K}$ , the convex QP problem is formulated as

$$\begin{aligned} \min_{\mathbf{p}} \quad & J_{ss} + \tau \sum_{k \in \mathcal{K}} l_{abs,x_{mu},k} + l_{abs,x_{lm},k} + l_{abs,v,k} \quad \text{with} \\ & (64a) \end{aligned}$$

$$\begin{aligned} \text{s.t.} \quad & l_{abs,x_{mu},k} := (x_{up,k} - x_{mid,k}) - \sigma_{x_{mu},k}(x_{low,k}^{\bar{}} - \bar{x}_{tr}) \\ & l_{abs,x_{lm},k} := (x_{low,k}^{\bar{}} - x_{low,k}) - \sigma_{x_{lm},k}(x_k - \underline{x}_{tr}) \\ & l_{abs,v,k} := (v_{pos,k} - v_{neg,k}) - \sigma_{v,k}v_k, \end{aligned} \quad (64b)$$

$$\mathbf{c}_{coll,i} \stackrel{(5)}{=} \mathbf{0} \quad \text{with} \quad \mathbf{f}_k = \begin{bmatrix} v_k \\ \frac{1}{m} (mg - F_{d,k}^{SSC} - F_{c,3,k}^{SSC}) \end{bmatrix},$$

$$F_{d,k}^{SSC} = u_{neg,k} + u_{pos,k} \quad \text{and}$$

$$F_{c,3,k}^{SSC} = c_{up}(x_{up,k} - \bar{x}_{tr}) + c_{mid}x_{mid,k} + c_{low}(x_{low,k} - \underline{x}_{tr}), \quad (64c)$$

$$\begin{aligned}
x_k \in \mathcal{X} &:= [\underline{x}, \bar{x}], v_k \in \mathcal{V} := [\underline{v}, \bar{v}], \\
u_{\text{neg},k} \in \mathcal{U}_{\text{neg}} &:= [F_d, 0], u_{\text{pos},k} \in \mathcal{U}_{\text{pos}} := [0, \bar{F}_d], \\
v_{\text{neg},k} \in \mathcal{V}_{\text{neg}} &:= [\underline{v}, 0], v_{\text{pos},k} \in \mathcal{V}_{\text{pos}} := [0, \bar{v}], \\
x_{\text{low},k} \in \mathcal{X}_{\text{low}} &:= [\underline{x}, \underline{x}_{\text{tr}}], x_{\text{low},k}^{\leftarrow} \in \mathcal{X}_{\text{low}}^{\leftarrow} := [\underline{x}_{\text{tr}}, \bar{x}], \\
x_{\text{mid},k} \in \mathcal{X}_{\text{mid}} &:= [\underline{x}_{\text{tr}}, \bar{x}_{\text{tr}}], x_{\text{up},k} \in \mathcal{X}_{\text{up}} := [\bar{x}_{\text{tr}}, \bar{x}]
\end{aligned} \tag{64d}$$

$$\begin{aligned}
\hat{\mathbf{h}}_{\text{zcs},p,k} &= \begin{bmatrix} -\bar{d} & 1 \\ \underline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\text{pos},k} \\ u_{\text{pos},k} \end{bmatrix} \leq \mathbf{0}, \\
\hat{\mathbf{h}}_{\text{zcs},n,k} &= \begin{bmatrix} -\bar{d} & 1 \\ \underline{d} & -1 \end{bmatrix} \begin{bmatrix} v_{\text{neg},k} \\ u_{\text{neg},k} \end{bmatrix} \leq \mathbf{0},
\end{aligned} \tag{64e}$$

$$\begin{aligned}
g_{\text{aff},x_{\text{mu}},k} &= x_{\text{mid},k} + x_{\text{up},k} - (x_{\text{low},k}^{\leftarrow} + \bar{x}_{\text{tr}}) = 0 \\
g_{\text{aff},x_{\text{lm}},k} &= x_{\text{low},k} + x_{\text{low},k}^{\leftarrow} - (x_k + \underline{x}_{\text{tr}}) = 0
\end{aligned} \tag{64f}$$

$$\begin{aligned}
g_{\text{aff},v,k} &= v_{\text{pos},k} + v_{\text{neg},k} - v_k = 0, \\
\underbrace{\begin{bmatrix} l_{\text{abs},x_{\text{mu}},k} | \sigma_{x_{\text{mu}},k} = +1 \\ l_{\text{abs},x_{\text{mu}},k} | \sigma_{x_{\text{mu}},k} = -1 \end{bmatrix}}_{=\mathbf{h}_{\text{abs},x_{\text{mu}},k}} &\geq \mathbf{0}, \quad \underbrace{\begin{bmatrix} l_{\text{abs},x_{\text{lm}},k} | \sigma_{x_{\text{lm}},k} = +1 \\ l_{\text{abs},x_{\text{lm}},k} | \sigma_{x_{\text{lm}},k} = -1 \end{bmatrix}}_{=\mathbf{h}_{\text{abs},x_{\text{lm}},k}} \geq \mathbf{0},
\end{aligned} \tag{64g}$$

$$\begin{aligned}
\underbrace{\begin{bmatrix} l_{\text{abs},v,k} | \sigma_{v,k} = +1 \\ l_{\text{abs},v,k} | \sigma_{v,k} = -1 \end{bmatrix}}_{=\mathbf{h}_{\text{abs},v,k}} &\geq \mathbf{0}, \\
\mathbf{u}_0 &= \begin{bmatrix} \max(dv_{\text{iv}}, 0) \\ \min(dv_{\text{iv}}, 0) \end{bmatrix}, \quad \mathbf{x}_0 = \mathbf{x}_{\text{iv}}.
\end{aligned} \tag{64h}$$

The termination criterion is computed via

$$\bar{l}_{\text{abs}} = \left\| \left[ \begin{array}{c} l_{\text{abs},x_{\text{mu}},0} \\ l_{\text{abs},x_{\text{lm}},0} \\ l_{\text{abs},v,0} \end{array} \right], \dots, \left[ \begin{array}{c} l_{\text{abs},x_{\text{mu}},n_{\text{seg}}} \\ l_{\text{abs},x_{\text{lm}},n_{\text{seg}}} \\ l_{\text{abs},v,n_{\text{seg}}} \end{array} \right] \right\|_{\max}. \tag{65}$$

In order to analyse the effect of the initialisation on the computed solution, several initial guesses are fed to the SSC algorithm aiming at solving the OP with initial value  $\mathbf{x}_{\text{iv},1}$  considering  $n_{\text{seg}} = 250$  collocation segments. The maximum number of iterations is increased to  $q_{\text{max}} = 25$  to check if poor approximations eventually converge. Using the solution of the NLP optimiser  $\mathbf{u}_{\text{opt}}^{\text{NLP}}$  and  $\mathbf{x}_{\text{opt}}^{\text{NLP}}$  as reference, the initial guesses

are generated using

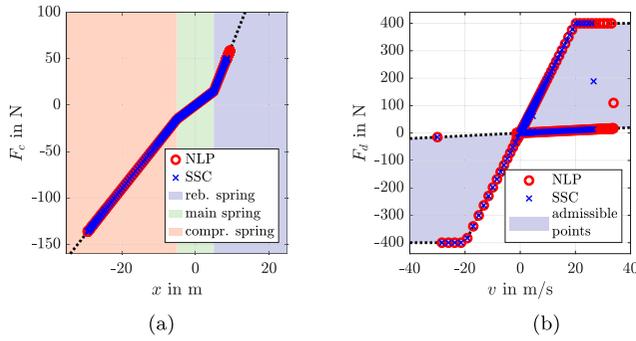
$$\begin{aligned}
u_{\text{neg},k}^* &= k_{\text{init}} \min(u_{\text{opt},k}^{\text{NLP}}, 0), u_{\text{pos},k}^* = k_{\text{init}} \max(u_{\text{opt},k}^{\text{NLP}}, 0), \\
\mathbf{x}_k^* &= k_{\text{init}} \mathbf{x}_{\text{opt},k}^{\text{NLP}}
\end{aligned} \tag{66}$$

with  $k_{\text{init}} \in \{0.0, 0.5, 0.75, 0.95, 1.0, 1.05, 10.0\}$  which are fed to the SSC update routine. The computations are compared to the results of the NLP problem which is initialised with the same guesses. The results of the individual optimisations are listed in Table 4. As Figure 8(a,b) indicates for  $k_{\text{init}} = 0$ , all solutions satisfy the prescribed constraints: The spring forces lie on the piecewise linear branches and the damper forces are within the admissible area. Considering the objective value  $J_{\text{ss}}$  and mean error  $\bar{e}_{\text{ss}}$  in Table 4, the NLP solver converges to the same solution independently which initial guess is supplied. Thus, it is assumed that the computed NLP solution represents the global optimum or is at least very close to it. However, initialisation can have a massive impact on the computation time. Generally, the guesses close to the solution result in shorter computation times. It is striking that both algorithms have the most difficulty converging for  $k_{\text{init}} = 0.5$  which is not the guess with the biggest difference to the optimal solution. The SSC algorithm converges significantly faster than the NLP solver for all initial guesses. The number of superordinate iterations required by the SSC algorithm is small for good initial guesses. As for  $k_{\text{init}} = 0.5$ , a high number of superordinate iterations can still yield a faster convergence compared to the NLP solver. As already mentioned, the SSC algorithm represents a local method and does not necessarily provide the global optimum, which is reflected in the results. Considering the objective value  $J_{\text{ss}}$  and mean error  $\bar{e}_{\text{ss}}$ , only the SSC solutions for  $k_{\text{init}} \geq 0.5$  are roughly in the vicinity of the global solution. The mean deviance between the input trajectories  $\bar{e}_{F_d}$  is then rather small. Even though the SSC solutions may not represent the global optimum, the solutions satisfy the prescribed constraints since  $q < q_{\text{max}}$ . Thus, they represent a local solution of the piecewise linear problem, which corresponds to the original problem for the example at hand. The results for  $k_{\text{init}} = \{0.95, 1.0, 1.05\}$  confirm that the

**Table 4.** Analysis of sensitivity regarding initial guess.

$k_{\text{init}}[-]$	method		results					
	NLP	SSC	$J_{\text{ss}}[\text{m}^2 \text{s}]$	$\bar{e}_{\text{ss}}[\text{m}]$	$\bar{e}_{F_d}[\text{N}]$	$q[-]$	$t_{\text{opt},\Sigma}[\text{s}]$	$t_{\Sigma}[\text{s}]$
0.0	✓		1603.95	5.47	–	1	4.589	4.589
0.0		✓	2118.49	9.91	46.63	7	0.493 (10.74%)	0.798 (17.38%)
0.5	✓		1603.95	5.47	–	1	6.347	6.347
0.5		✓	1635.20	5.93	14.43	19	1.450 (22.85%)	2.136 (33.65%)
0.75	✓		1603.95	5.47	–	1	0.557	0.557
0.75		✓	1612.62	5.67	15.77	2	0.179 (32.14%)	0.324 (58.26%)
0.95	✓		1603.95	5.47	–	1	0.473	0.473
0.95		✓	1603.95	5.47	0.01	1	0.075 (15.86%)	0.199 (42.02%)
1.0	✓		1603.95	5.47	–	1	0.417	0.417
1.0		✓	1603.95	5.47	0.01	1	0.072 (17.28%)	0.191 (45.79%)
1.05	✓		1603.95	5.47	–	1	0.427	0.427
1.05		✓	1603.96	5.47	0.20	1	0.084 (19.67%)	0.202 (47.37%)
10.0	✓		1603.95	5.47	–	1	0.981	0.981
10.0		✓	1610.74	5.72	6.71	2	0.149 (15.19%)	0.306 (31.22%)

$k_{\text{init}}$ : perturbation parameter for initialisation error; NLP: IPOPT solving NLP problem; SSC: SSC Algorithm 1 with QP problem (64a);  $J_{\text{ss}}$ : main objective (48);  $\bar{e}_{\text{ss}}/\bar{e}_{F_d}$ : error metrics (58) and (57);  $q$ : # of superordinate iterations;  $t_{\text{opt},\Sigma}$ : cumulative time spent in NLP/QP solver;  $t_{\Sigma}$ : total computation time.



**Figure 8.** Optimisation results for SMO application with three-segmented nonlinear spring for  $k_{\text{init}} = 0$ . (a) Piecewise linear spring force with three segments. (b) ZCS for damper force.

SSC algorithm converges to the global optimum if the initial guess is in a close vicinity of it.

As mentioned in Section 2.4, increasing the number of space divisions to depict the three-segmented spring raises computation time. This can be confirmed by comparing solution times of the SSC algorithm for  $k_{\text{init}} = 0.0$ : For the two-segmented spring  $t_{\text{opt},\Sigma}^{\text{SSC}} = 0.262\text{s}$ ,  $t_{\Sigma}^{\text{SSC}} = 0.329\text{s}$  and  $q = 4$  holds whereas  $t_{\text{opt},\Sigma}^{\text{SSC}} = 0.493\text{s}$ ,  $t_{\Sigma}^{\text{SSC}} = 0.798\text{s}$  and  $q = 7$  holds for the spring with three segments. Thus, more iterations are required and the average computation time per iteration is increased by 7.52% and 38.6% for  $t_{\text{opt},\Sigma}^{\text{SSC}}$  and  $t_{\Sigma}^{\text{SSC}}$ , respectively. This highlights the importance of an implementation in C-Code to reduce the time required for the updating routine and therefore the overall computation time.

### 3.4 Rotated space splitting

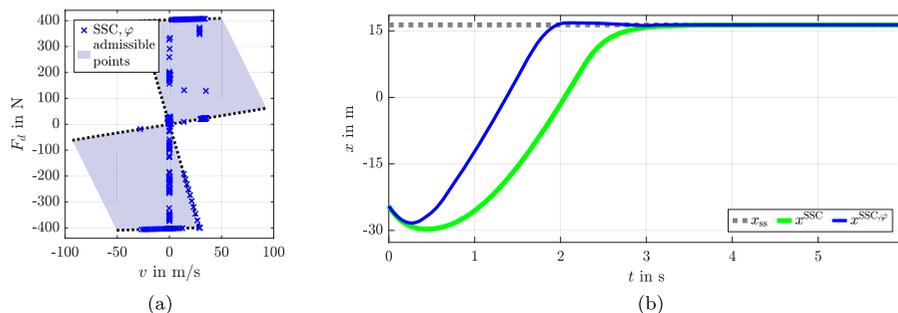
As elaborated in Section 2.3.1, some sets require to be split using a rotated straight line. Without going into detail, the approach is applied to the example of a hanging SMO with fully linear spring presented in Section 3.1. The OP is parametrised with initial value  $x_{\text{iv},1}$  and  $n_{\text{seg}} = 250$  collocation segments. The damper set is rotated by  $\varphi = 7^\circ$  resulting in the set depicted in Figure 9(a). This is implemented by applying the changes (36) to the convex QP problem with fully linear spring. As depicted in Figure 9(a), the SSC algorithm computes damper forces which all lie within the admissible area. Although this example represents an artificial use-case, the corresponding position trajectories in Figure 9(b) prove that the rotated set yields a reduction of the objective. Starting with  $x_{\text{iv},1} < x_{\text{ss}}$  in a compression phase

with  $v_{\text{iv},1} < 0$ , the mass is first decelerated and then accelerated using  $F_d < 0$  at  $v > 0$ , hence active forces. These active forces enable a faster attainment of the steady-state position. Due to the similarity to the previous examples, no further discussion is given.

## 4. Conclusions

This article has presented a novel local solution method for efficiently solving nonconvex optimal control problems. Splitting the space of inputs and states enables the formulation of QP sub-problems which are solved iteratively. The method is capable of considering two types of nonconvexities: ZCSs and equality constraints with possibly multiple univariate nonlinearities. The approach decomposes the nonconvexities into affine parts which are linked appropriately to approximate the original problem with scalable approximation error. It has been shown that the SSC algorithm converges to a local optimum of this approximated problem. The quality of the solution depends on the initial guess. For good initial guesses, the method converges to solutions close to the global one. A comparison of this successive convexification technique with other existing methods has been given, highlighting the advantages SSC provides.

Subsequently, possible directions for future research are disclosed. Even though it would not change the algorithm, using nondifferentiable constraint qualifications (Ye, 2004) could potentially avoid the AVC-smoothing required for continuous differentiability regarding the convergence proof. Due to the low computation time and robust convergence, the SSC method seems to be a promising approach for real-time optimal control applications. In this context, using MPC could provide sufficiently good initial guesses (Diehl et al., 2005; Gros et al., 2020) resulting in good SSC solutions. In order to enhance applicability, the SSC approach could be combined with other iterative linearisation schemes like (Carvalho et al., 2013; Liniger et al., 2015; Mao et al., 2017; Simon et al., 2013) which could enable considering collision avoidance constraints. Furthermore, extending the SSC approach to bivariate nonlinearities in equality constraints would enlarge the possible scope of applications. Moreover, applying a more sophisticated updating procedure for the penalty parameter would improve computation times (Maratos, 1978; Mayne & Maratos, 1979; Nocedal & Wright, 2006). For applications with real-time capability requirement, it is recommended to implement the algorithm via tailored C-code to avoid the overhead introduced by the



**Figure 9.** SSC optimisation results for SMO application with fully linear spring and rotated damper set. (a) Rotated ZCS for damper force. (b) Position trajectory;  $x^{\text{SSC}}$ : original damper force set;  $x^{\text{SSC},\varphi}$ : rotated damper force set.

high-level routines of *JuMP* and the *Gurobi*-wrapper. Then, the required time for the updating step can be greatly reduced to linear time complexity resulting in shorter overall computation times.

## Notation

$\hat{(\cdot)}$	Transformed constraints depending on $\mathbf{z}$
$\hat{(\cdot)}$	Projected initial solution of $(\cdot)$
$(\cdot)^*$	Previous optimal solution of $(\cdot)$
$(\cdot)_{\text{tr}}$	Transition point for space bisection
$(\cdot), (\cdot)$	Lower and upper bound of $(\cdot)$
$(\cdot)_{n/\text{neg}}$	Negative region of bisected space
$(\cdot)_{p/\text{pos}}$	Positive region of bisected space
$\alpha(\cdot)$	Smoothable absolute value function
$\hat{\mathbf{A}}, \hat{\mathbf{b}}$	Matrix and vector of affine transformation function
$\mathbf{A}, \mathbf{B}, \mathbf{k}$	Matrices and vector for system behaviour description
$\mathbf{A}_{\text{ZCS}}, \mathbf{b}_{\text{ZCS}}$	Matrix and vector describing affine ZCSs
$c(\cdot)$	Spring stiffness of corresponding segment $(\cdot)$ for SMO
$\mathbf{c}$	Left-hand sides of equality constraints
$\tilde{\mathbf{c}}$	Left-hand sides of continuous-time equality constraints
$\mathbf{c}_{\text{coll}}$	Left-hand sides of collocation equality constraints
$\mathbf{c}_{\text{tot}}$	Left-hand sides of aggregated equality constraints
$d$	Damper coefficient for SMO
$\Delta_i$	Width of $i^{\text{th}}$ collocation segment
$\bar{e}_{F_d}$	Mean absolute deviance in damper force trajectories between NLP and SSC solution
$e_{\text{ss}}$	Steady-state position deviation of SMO
$\bar{e}_{\text{ss}}$	Mean absolute deviance between position trajectory and steady-state position
$\varepsilon_g$	Error tolerance for linearised AVCs
$\varepsilon_s$	Smoothing parameter for absolute value function
$\mathbf{E}_{\Sigma/\Delta/z}$	Matrices for selecting specific decision variables
$\mathbf{f}$	Right-hand side of differential equation system of model
$\mathbf{f}_{\Phi}$	Affine approximation of right-hand side of model equations
$\mathbf{f}_{\text{pwl}}$	Composition of univariate nonlinearities approximable by piecewise linear curves
$F_c$	Spring force of SMO
$F_d$	Damper force of SMO
$\phi$	Intermediate objective term
$\Phi_{x/u/\text{pwl}}$	Mapping functions affine in the decision variables
$g$	Gravitational acceleration for SMO
$\mathbf{g}_{\text{abs}}$	Left-hand sides of AVCs for space splitting
$\mathbf{g}_{\text{aff}}$	Left-hand sides of affine constraints for space splitting
$\mathbf{h}$	Left-hand sides of inequality constraints
$\tilde{\mathbf{h}}$	Left-hand sides of continuous-time inequality constraints
$\mathbf{h}_{\text{abs}}$	Left-hand sides of relaxed (inequality) AVCs for space splitting
$\mathbf{h}_{\text{tot}}$	Left-hand sides of aggregated inequality constraints

$\mathbf{h}_{\text{ZCS}}$	Left-hand sides of ZCS inequality constraints
$\mathcal{I}$	Set of collocation segment indices
$J$	Objective function
$\tilde{J}$	Continuous-time objective functional
$J_{\text{ss}}$	Objective function for steady-state position deviation of SMO
$k_{\text{init}}$	Perturbation parameter for initialisation error
$\mathcal{K}$	Set of collocation point indices
$\mathbf{l}_{\text{abs}}$	Left-hand sides of linearised AVCs for space splitting
$m$	Mass of SMO
$n_c$	Number of equality constraints
$n_h$	Number of inequality constraints
$n_{\text{pts}}$	Number of collocation points
$n_s$	Number of auxiliary decision variables
$n_{\text{seg}}$	Number of collocation segments
$n_u$	Number of input decision variables
$n_w$	Number of original decision variables
$n_x$	Number of state decision variables
$\mathcal{O}(\cdot)$	Landau-notation for computational complexity
$\mathbf{p}_0, \mathbf{q}_0, r_0$	parameters of quadratic objective
$q$	Number of superordinate iterations in SSC algorithm
$q_{\text{max}}$	Maximum number of superordinate iterations
$\mathbb{R}^n$	Set of $n$ -dimensional, real vectors
$s_w$	Slack variable for AVC
$\mathcal{S}(\cdot)$	Set described by inequality constraints
$\sigma(\cdot)$	Sign of corresponding AVC
$\Sigma$	Matrix containing all linearisation points of AVCs
$t$	Time
$t_{\text{opt}, \Sigma}$	Cumulative time spent in solver
$t_{\Sigma}$	Total computation time
$\tau$	Penalty parameter for additional objective terms
$\vartheta$	Terminal objective term
$\mathbf{u}$	Input vector of system
$\mathcal{U}$	Bounding-box set for system inputs
$\mathbf{U}_0^*$	Initial solution for input decision variables
$v$	Deflection velocity of SMO
$\mathbf{w}$	Vector of original decision variables
$\mathbf{w}_{\text{aux}}$	Vector of auxiliary decision variables
$w_{\text{low/up}}$	Lower/upper region of bisected space of $w$
$x$	Deflection position of SMO
$x_{\text{ss}}$	Steady-state position of SMO
$x_{\text{low/low}}$	Lower/upper region of position space bisection for SMO
$x_{\text{mid/up}}$	Lower/upper region of bisected position space $x_{\text{low}}$ for SMO
$\mathbf{x}$	State vector of system
$\mathcal{X}$	Bounding-box set for system states
$\mathbf{X}_0^*$	Initial solution for state decision variables
$\mathbf{x}_{\text{iv}}$	Initial value of system state for SMO
$\mathbf{z}$	Vector of accumulated decision variables
$\tilde{\mathbf{z}}$	Argument vector for absolute value function
$\tilde{\mathbf{z}}^*$	Linearisation point for AVCs

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## ORCID

Tadeas Sedlacek  <http://orcid.org/0000-0002-6191-6173>

Dirk Odenthal  <http://orcid.org/0000-0002-6651-8369>

Dirk Wollherr  <http://orcid.org/0000-0003-2810-6790>

## References

- Acikmese, B., & Blackmore, L. (2011). Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2), 341–347. <https://doi.org/10.1016/j.automatica.2010.10.037>
- Acikmese, B., & Ploen, S. R. (2007). Convex programming approach to powered descent guidance for Mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5), 1353–1366. <https://doi.org/10.2514/1.27553>
- Armaou, A., & Ataei, A. (2014). Piece-wise constant predictive feedback control of nonlinear systems. *Journal of Process Control*, 24(4), 326–335. <https://doi.org/10.1016/j.jprocont.2014.02.002>
- Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Society for Industrial and Applied Mathematics.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization* (7th ed.). Cambridge University Press.
- Carvalho, A., Gao, Y., Gray, A., Tseng, H. E., & Borrelli, F. (2013). *Predictive control of an autonomous ground vehicle using an iterative linearization approach*. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013) (p. 2335–2340). doi:10.1109/ITSC.2013.6728576
- Chen, J., Zhan, W., & Tomizuka, M. (2017). *Constrained iterative LQR for on-road autonomous driving motion planning*. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (p. 1–7). doi:10.1109/ITSC.2017.8317745
- Cibulka, V., Hanis, T., & Hromcik, M. (2019). *Data-driven identification of vehicle dynamics using Koopman operator*. <https://arxiv.org/abs/1903.06103>
- Cibulka, V., Haniš, T., Korda, M., & Hromčík, M. (2020). Model predictive control of a vehicle using Koopman operator. *IFAC-PapersOnLine*, 53(2), 4228–4233. <https://doi.org/10.1016/j.ifacol.2020.12.2469> (21st IFAC World Congress)
- d'Aspremont, A., & Boyd, S. (2003). *Relaxations and randomized methods for nonconvex QCQPs*. Retrieved November 24, Retrieved 2020, from <https://web.stanford.edu/class/ee392o/relaxations.pdf>
- Del Re, L., Chapuis, J., & Nevistic, V. (1993). *Predictive control with embedded feedback linearization for bilinear plants with input constraints*. Proceedings of 32nd IEEE Conference on Decision and Control (Vol. 4, p. 2984–2989). doi:10.1109/CDC.1993.325747
- Diehl, M., Bock, H. G., & Schlöder, J. P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736. <https://doi.org/10.1137/S0363012902400713>
- Di Pillo, G., & Grippo, L. (1989). Exact penalty functions in constrained optimization. *SIAM Journal on Control and Optimization*, 27(6), 1333–1360. <https://doi.org/10.1137/0327068>
- Dunning, I., Huchette, J., & Lubin, M. (2017). JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2), 295–320. <https://doi.org/10.1137/15M1020575>
- Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., & Hrovat, D. (2007). Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3), 566–580. <https://doi.org/10.1109/TCST.2007.894653>
- Fang, Y., & Armaou, A. (2015). *Nonlinear model predictive control using a bilinear Carleman linearization-based formulation for chemical processes*. 2015 American Control Conference (ACC) (p. 5629–5634). doi:10.1109/ACC.2015.7172221
- Ferreau, H., Almér, S., Verschueren, R., Diehl, M., Frick, D., Domahidi, A., Jerez, J.L., Stathopoulos, G., & Jones, C. (2017). Embedded optimization methods for industrial automatic control. *IFAC-PapersOnLine*, 50(1), 13194–13209. <https://doi.org/10.1016/j.ifacol.2017.08.1946>
- Gros, S., Zanon, M., Quirynen, R., Bemporad, A., & Diehl, M. (2020). From linear to nonlinear MPC: Bridging the gap via the real-time iteration. *International Journal of Control*, 93(1), 62–80. <https://doi.org/10.1080/00207179.2016.1222553>
- Gurobi Optimization (2020). *Gurobi optimizer reference manual*. Retrieved November 24, 2020, from [https://www.gurobi.com/wp-content/plugins/hd\\_documentations/documentation/9.1/refman.pdf](https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.1/refman.pdf)
- Han, S. P., & Mangasarian, O. L. (1979). Exact penalty functions in nonlinear programming. *Mathematical programming*, 17(1), 251–269. <https://doi.org/10.1007/BF01588250>
- Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4, 373–395. <https://doi.org/10.1007/BF02579150>
- Kelly, M. (2017). An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4), 849–904. <https://doi.org/10.1137/16M1062569>
- Khalil, H. K. (2002). *Nonlinear systems* (3rd ed.). Prentice Hall.
- Korda, M., & Mezić, I. (2018). Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93(1), 149–160. <https://doi.org/10.1016/j.automatica.2018.03.046>
- Kozlov, M., Tarasov, S., & Khachiyan, L. (1980). The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5), 223–228. [https://doi.org/10.1016/0041-5553\(80\)90098-1](https://doi.org/10.1016/0041-5553(80)90098-1)
- Kurtz, M. J., & Henson, M. A. (1997). Input-output linearizing control of constrained nonlinear processes. *Journal of Process Control*, 7(1), 3–17. [https://doi.org/10.1016/S0959-1524\(96\)00006-6](https://doi.org/10.1016/S0959-1524(96)00006-6)
- Liberti, L. (2008). *Introduction to global optimization*. Retrieved January 23, 2021, from <https://www.lix.polytechnique.fr/liberti/teaching/globalopt-lima.pdf>
- Liberti, L., & Maculan, N. (2006). *Global optimization – from theory to implementation*. Springer Science and Business Media.
- Liniger, A., Domahidi, A., & Morari, M. (2015). Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36, 628–647. <https://doi.org/10.1002/oca.2123>
- Lipp, T., & Boyd, S. (2016). Variations and extension of the convex-concave procedure. *Optimization and Engineering*, 17, 263–287. <https://doi.org/10.1007/s11081-015-9294-x>
- Mangasarian, O. L. (2015). A hybrid algorithm for solving the absolute value equation. *Optimization Letters*, 9, 1469–1474. <https://doi.org/10.1007/s11590-015-0893-4>
- Mao, Y., Dueri, D., Szmuk, M., & Açikmeşe, B. (2017). Successive Convexification of Non-Convex Optimal Control Problems with State Constraints. *IFAC-PapersOnLine*, 50(1), 4063–4069. <https://doi.org/10.1016/j.ifacol.2017.08.789> (20th IFAC World Congress)
- Mao, Y., Szmuk, M., & Açikmeşe, B. (2016). *Successive convexification of non-convex optimal control problems and its convergence properties*. 2016 IEEE 55th Conference on Decision and Control (CDC) (p. 3636–3641). <https://doi.org/10.1109/CDC.2016.7798816>
- Maratos, N. (1978). *Exact penalty function algorithms for finite dimensional and control optimization problems* [Unpublished doctoral dissertation]. Imperial College London (University of London).
- Mayne, D. Q. (1973). Differential dynamic programming – A unified approach to the optimization of dynamic systems. In C. Leondes (Ed.), *Control and dynamic systems* (Vol. 10, p. 179–254). Academic Press.
- Mayne, D. Q., & Maratos, N. (1979). A first order, exact penalty function algorithm for equality constrained optimization problems. *Mathematical Programming*, 16(1), 303–324. <https://doi.org/10.1007/BF01582118>
- Mollov, S., Babuska, R., Abonyi, J., & Verbruggen, H. B. (2004). Effective optimization for fuzzy model predictive control. *IEEE Transactions on Fuzzy Systems*, 12(5), 661–675. <https://doi.org/10.1109/TFUZZ.2004.834812>
- Mollov, S., Babuška, R., Roubos, J., & Verbruggen, H. (1998). MIMO predictive control by multiple-step linearization of Takagi-Sugeno fuzzy models. *IFAC Proceedings Volumes*, 31(29), 197–202. [https://doi.org/10.1016/S1474-6670\(17\)38944-9](https://doi.org/10.1016/S1474-6670(17)38944-9) (7th IFAC Symposium on Artificial Intelligence in Real Time Control 1998., Grand Canyon National Park, USA, 5–8 October)
- Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science and Business Media.
- Park, J., & Boyd, S. (2017). *General heuristics for nonconvex quadratically constrained quadratic programming*. <https://arxiv.org/abs/1703.07870>

- Perantoni, G., & Limebeer, D. J. (2014). Optimal control for a Formula One car with variable parameters. *Vehicle System Dynamics*, 52(5), 653–678. <https://doi.org/10.1080/00423114.2014.889315>
- Raković, S. V., & Levine, W. S. (2018). *Handbook of model predictive control*. Springer.
- Ryoo, H., & Sahinidis, N. (1995). Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers and Chemical Engineering*, 19(5), 551–566. [https://doi.org/10.1016/0098-1354\(94\)00097-2](https://doi.org/10.1016/0098-1354(94)00097-2)
- Savaresi, S. M., Poussot-Vassal, C., Spelta, C., Sename, O., & Dugard, L. (2010). *Semi-active suspension control design for vehicles* (1st ed.). Elsevier.
- Schnelle, F., & Eberhard, P. (2015). Constraint mapping in a feedback linearization/MPC scheme for trajectory tracking of underactuated multibody systems. *IFAC-PapersOnLine*, 48(23), 446–451. <https://doi.org/10.1016/j.ifacol.2015.11.319> (5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015)
- Sedlacek, T., Odenthal, D., & Wollherr, D. (2020a). Convexification of semi-activity constraints applied to minimum-time optimal control for vehicles with semi-active limited-slip differential. In *17th International Conference on Informatics in Control, Automation and Robotics – volume 1: Icinco* (p. 15–25). SciTePress.
- Sedlacek, T., Odenthal, D., & Wollherr, D. (2020b). Minimum-time optimal control for battery electric vehicles with four wheel-independent drives considering electrical overloading. *Vehicle System Dynamics*, 1–25. <https://doi.org/10.1080/00423114.2020.1823004>
- Sedlacek, T., Odenthal, D., & Wollherr, D. (2020c). Minimum-time optimal control for vehicles with active rear-axle steering, transfer case and variable parameters. *Vehicle System Dynamics*, 59(8), 1227–1255. <https://doi.org/10.1080/00423114.2020.1742925>
- Seki, H., Ooyama, S., & Ogawa, M. (2004). Nonlinear model predictive control using successive linearization. *Transactions of the Society of Instrument and Control Engineers, E-3*(1), 66–72. <https://www.sice.jp/e-trans/papers/E3-9.pdf>
- Simon, D., Löfberg, J., & Glad, T. (2013). *Nonlinear model predictive control using feedback linearization and local inner convex constraint approximations*. 2013 European Control Conference (ECC) (p. 2056–2061). <https://doi.org/10.23919/ECC.2013.6669575>
- Tanaka, K., & Wang, H. O. (2004). *Fuzzy control systems design and analysis – a linear matrix inequality approach*. John Wiley and Sons.
- Tassa, Y., Erez, T., & Todorov, E. (2012). *Synthesis and stabilization of complex behaviors through online trajectory optimization*. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (p. 4906–4913). <https://doi.org/10.1109/IROS.2012.6386025>
- Tassa, Y., Mansard, N., & Todorov, E. (2014). *Control-limited differential dynamic programming*. 2014 IEEE International Conference on Robotics and Automation (ICRA) (p. 1168–1175). <https://doi.org/10.1109/ICRA.2014.6907001>
- Tawarmalani, M., & Sahinidis, N. V. (2002). *Convexification and global optimization in continuous and mixed-integer nonlinear programming – theory, algorithms, software, and applications*. Springer Science and Business Media.
- Verschueren, R. (2018). *Convex approximation methods for nonlinear model predictive control* [Unpublished doctoral dissertation]. University of Freiburg.
- Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57. <https://doi.org/10.1007/s10107-004-0559-y>
- Xie, Z., Liu, C. K., & Hauser, K. (2017). *Differential dynamic programming with nonlinear constraints*. 2017 IEEE International Conference on Robotics and Automation (ICRA) (p. 695–702). <https://doi.org/10.1109/ICRA.2017.7989086>
- Ye, J. J. (2004). Nondifferentiable multiplier rules for optimization and Bilevel optimization problems. *SIAM Journal on Optimization*, 15(1), 252–274. <https://doi.org/10.1137/S1052623403424193>
- Zorich, V. A. (2016). *Mathematical analysis I*. Springer.