The Crawling Phenomenon in Sequential Convex Programming

Taylor P. Reynolds and Mehran Mesbahi

Abstract— **The paper examines the so-called crawling phenomenon for a class of sequential convex programming algorithms. These algorithms are designed to solve non-convex optimization problems by using convex approximations, trust regions and relaxations. The crawling phenomenon occurs when the iterates of the algorithm get as close as permitted to each other, yet these iterates are not close to a stationary point of the original non-convex problem. It is shown that once the design parameters for a general class of such iterative algorithms are fixed, there are generic problem instances that lead to the crawling phenomenon. A simple example and potential remedies to address this phenomenon are also presented.**

Index Terms— **Non-convex optimization; sequential convex programming; crawling phenomenon**

I. INTRODUCTION

Solving non-convex optimization problems is difficult. Alas, non-convex problems are pervasive in control synthesis problems, and more generally, engineering system design. Examples include optimal control problems with nonlinear dynamics (represented as non-convex algebraic or differential equality constraints), or problems that have non-convex constraints that cannot be "losslessly convexified" [1], [2], [3]. These problems are found in fields ranging from aerospace guidance [4], [5], [6] and mechanical truss design [7] to power grid optimization [8], and computer vision [9]. For such non-convex optimization problems, sequential convex programming (SCP) is a powerful framework with which one can design algorithms that find the desired solution(s). Other such techniques include nonlinear programming [10], [11], sum of squares optimization [12], [13] and evolutionary algorithms.

Sequential convex programming is a natural approach for solving non-convex optimization problems; convex programming is generally thought of as "easy" (from a computational perspective due to the availability of interior point methods) and the associated theoretical backing (such as strong duality) is well established [14], [15]. Different variations of SCP techniques stem from the same idea: solve a sequence of convex approximations to the original non-convex problem, each time using the solution of a previous iteration's convex problem to improve the approximation. The main challenge lies in *how* the convex approximations are formulated, what structure is devised for measuring progress towards an optimal solution and updating the approximations, and how all of this lends itself to theoretical analysis.

This research has been supported by NASA grant NNX17AH02A SUP02 and NSERC grant PGSD3-502758-2017.

The authors are with the W.E. Boeing Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, USA. {tpr6,mesbahi}@uw.edu

Likely the simplest class of SCP methods is that of sequential linear programming (SLP). Algorithms in this class linearize all nonlinear functions about a current reference solution to obtain a linear program. The linear program is then solved (with a trust region) to obtain a new reference, and the process repeats [16], [17]. From a computational perspective, SLP became attractive early due to the maturity of the simplex algorithm. However, over time solvers for more general classes of convex optimization problems have advanced to the point that restricting oneself to linear programs to save computational resources is unnecessary (except perhaps for extremely large problems). More generally, difference of convex functions, or D.C. programming, is a class of SCP methods [18]. The convex-concave procedure introduced in [19] decomposes each non-convex inequality constraint into a sum of convex and concave functions, linearizing the concave part while leaving the convex part intact. This procedure remains popular in modern applications, in particular for support vector machines and principal component analysis in the field of machine learning [20].

Another important class of SCP methods is that of sequential quadratic programming (SQP). The collective works of Han [21], Powell [22], [23], Boggs and Tolle [24], [25] and Fukushima [26] exerted significant influence on the early developments of SQP algorithms, and their impact remains evident today. SQP methods approximate a non-convex problem with a quadratic program by approximating the Hessian of the non-convex problem's Lagrangian at a reference solution. The quadratic program is solved to obtain a new reference solution, and the process repeats. SQP methods are arguably among the most mature class of SCP methods [27], [28].

The use of quadratic programs requires that all constraints are affine in the solution variable. Many problems of interest are, however, subject to nonlinear constraints (both convex and non-convex). The class of SCP methods discussed herein are therefore those that solve a more general convex problem at each iteration (i.e., no a priori restriction to an LP or QP). More importantly, we discuss algorithms that are of trust region type and use slack variables to ensure that each convex approximation is always feasible. This class of SCP methods has been developed largely over the last decade, and represents one of the most active areas of current development [29], [30], [31], [32]. A subset of SLP and SQP methods are contained within this class, and our discussion holds for them as well.

The purpose of this paper is to demonstrate that this general class of SCP algorithms is susceptible to what we call the *crawling phenomenon*. The crawling phenomenon is defined as slow progress towards a stationary point of the non-convex problem when the algorithm is not close to any such solution. We show that the crawling phenomenon is a generic property of algorithms that use a Lagrangian-like function to measure the accuracy of the convex approximations at each iteration and update the trust region and reference solution accordingly. Essentially, the trust region and solution update rules that form a key component of the theoretical convergence analysis induce the undesirable crawling phenomenon. Arriving at a local minimum indeed implies that adjacent iterates will be close together (in the sense of their normed difference), but what we show is that for a certain general class of SCP algorithms, the converse does not necessarily hold.

This paper is organized as follows. First, we describe the generic class of SCP algorithms, our nomenclature and notation in §II. In §III we define the crawling phenomenon and prove that algorithms in this class are susceptible to it, followed by a simple example. Potential remedies for avoiding the crawling phenomenon are discussed in §IV, and §V offers concluding remarks.

II. SEQUENTIAL CONVEX PROGRAMMING

Consider the class of optimization problems of the form,

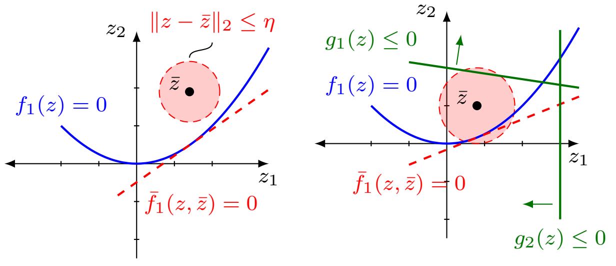
$$\begin{aligned} \min_{z} \quad & \phi(z) \\ \text{s.t.} \quad & g_i(z) \le 0, \qquad i \in \mathcal{I}_{\text{cvx}} \\ & h_i(z) \le 0, \qquad i \in \mathcal{I}_{\text{ncvx}} \\ & f_j(z) = 0, \qquad j \in \mathcal{E}_{\text{ncvx}} \end{aligned} \tag{$\mathcal{P}$}$$

where $z \in \mathbb{R}^{n_z}$, $\mathcal{I}_{\text{cvx}} = \{1,\dots,m_i\}$, $\mathcal{I}_{\text{ncvx}} = \{m_i + 1,\dots,m_i+n_i\}$ represent the indices of the convex and non-convex inequality constraints and $\mathcal{E}_{\text{ncvx}} = \{1,\dots,n_e\}$ represents the indices of non-convex equality constraints. We assume that each h_i and f_j are at least once differentiable and, without loss of generality, that the cost function ϕ is convex. To simplify the notation, we assume that any convex (affine) equality constraints are represented by a pair of convex inequality constraints. Problem $\mathcal{P}$ is typically referred to as a nonlinear programming problem, but we shall refer to it as the "original" problem. While there exist general purpose solvers that are able to solve the original problem (see, e.g., [10], [11]), we focus on iterative schemes that are based on convex optimization.

At each iteration, an SCP method approximates the non-convex constraints h_i and f_j with convex functions of the solution variable of the form

$$\bar{h}_i(z,\bar{z}) \le 0, \quad i \in \mathcal{I}_{\text{ncvx}}, \tag{1a}$$
$$\bar{f}_j(z,\bar{z}) = 0, \quad j \in \mathcal{E}_{\text{ncvx}}, \tag{1b}$$

where $\bar{z} \in \mathbb{R}^{n_z}$ is some reference. There are several choices for $\bar{h}_i$: first-order Taylor series, second-order Taylor series with (possibly approximated) positive semi-definite Hessian, inner convex approximations, or any other convex function that locally approximates the non-convex h_i. The $\bar{f}_j$ on the other hand must be affine functions of z. The approximations $\bar{h}_i$ and $\bar{f}_j$ are therefore local; their accuracy decreases as



(a) $\mathcal{F}_1 = \emptyset$, see (4a). (b) $\mathcal{F}_2 = \emptyset$, see (4b).

Fig. 1: A depiction of the two causes of artificial infeasibility.

$\|z - \bar{z}\|$ increases. As a separate issue, consider the possible scenario where ϕ and each g_i are linear functions, and we choose each $\bar{h}_i$ to be the linearization of h_i around $\bar{z}$. In this case, the resulting optimization problem is unbounded below, a phenomenon referred to as *artificial unboundedness*. To address the local nature of the approximations (1) and to avoid artificial unboundedness, we add a trust region constraint of the form,

$$\|z-\bar{z}\|_q \le \eta, \qquad q = \{1,\,2,\,\infty\}, \tag{2}$$

where $\eta \in \mathbb{R}_{++}$ is a positive trust region radius. We may then construct the following convex approximation to the original problem Problem $\mathcal{P}$:

$$\min_{z} \quad \phi(z) \tag{3a}$$
$$\text{s.t.} \quad g_i(z) \le 0, \qquad i \in \mathcal{I}_{\text{cvx}} \tag{3b}$$
$$\bar{h}_i(z,\bar{z}) \le 0, \qquad i \in \mathcal{I}_{\text{ncvx}} \tag{3c}$$
$$\bar{f}_j(z,\bar{z}) = 0, \qquad j \in \mathcal{E}_{\text{ncvx}} \tag{3d}$$
$$\|z-\bar{z}\|_q \le \eta. \tag{3e}$$

Problem 3 is not necessarily well-defined due to *artificial infeasibility* that can arise in two forms. The following independent cases, depicted in Figure 1, can happen

$$\mathcal{F}_1 = \{z \,|\, \text{(3c), (3d) and (3e) are satisfied}\} = \emptyset, \tag{4a}$$
$$\mathcal{F}_2 = \{z \,|\, \text{(3b), (3c) and (3d) are satisfied}\} = \emptyset. \tag{4b}$$

Artificial infeasibility can be avoided by adding so-called *virtual control* to (3c) and (3d) as,

$$\bar{h}_i(z,\bar{z}) - \sigma_i \le 0, \quad \sigma_i \ge 0 \tag{5a}$$
$$\bar{f}_j(z,\bar{z}) - \nu_j = 0. \tag{5b}$$

The vectors $\sigma \in \mathbb{R}^{n_i}_{+}$ and $\nu \in \mathbb{R}^{n_e}$ are added as solution variables, and the resulting augmented convex program assumes the form,

$$\begin{aligned} \min_{z,\sigma,\nu} \quad & \phi(z) + \lambda P(\sigma,\nu) \\ \text{s.t.} \quad & g_i(z) \le 0, \qquad i \in \mathcal{I}_{\text{cvx}} \\ & \bar{h}_i(z,\bar{z}) - \sigma_i \le 0, \qquad i \in \mathcal{I}_{\text{ncvx}} \\ & \bar{f}_j(z,\bar{z}) - \nu_j = 0, \qquad j \in \mathcal{E}_{\text{ncvx}} \\ & \|z-\bar{z}\|_q \le \eta, \end{aligned} \tag{$\mathcal{C}$}$$

where $P : \mathbb{R}^{n_e} \times \mathbb{R}^{n_i} \to \mathbb{R}_{++}$ is an exact positive-definite penalty function and $\lambda > 0$ is a user-selected weight.

Typically $\lambda \gg 1$ to limit the use of virtual control only to cases when it is required to avoid artificial infeasibility.

We shall refer to Problem $\mathcal{C}$ as a convex subproblem, and its optimal solution as an *iterate*, denoted by (z^*, σ^*, ν^*). We focus on the class of SCP algorithms that iteratively solve Problem $\mathcal{C}$ and use the resulting iterate to update $\bar{z}$ and the value of η, based on some assessment of the accuracy of the approximations (1). The resulting sequence of solutions $\bar{z}$ approaches a locally optimal solution of Problem $\mathcal{P}$ under certain conditions and assumptions. There have been numerous papers that study convergence properties of the such a setup, offering conditions/assumptions under which convergence is guaranteed to various definitions of minima [29], [30], [31], [32]. This paper does not discuss whether convergence is achieved or not, but instead demonstrates how the crawling phenomenon can affect the iterates as they converge.

A. Trust Region Updates

Most trust region methods dynamically update the value of η at each iteration according to a set of rules. These rules are designed to improve convergence and facilitate formal proofs, and use a performance metric that is a function of (z^*, σ^*, ν^*) to do so. The performance metric provides key feedback to the iterative process and guides convergence. To introduce two common performance metrics, we first define the functions,

$$J(z) = \phi(z) + \lambda P\big(h(z),\, f(z)\big), \tag{6a}$$

$$L(z) = \phi(z) + \lambda P\big(\bar{h}(z,\bar{z}),\, \bar{f}(z,\bar{z})\big), \tag{6b}$$

where $h(z) \in \mathbb{R}^{n_i}$ and $f(z) \in \mathbb{R}^{n_e}$ are the concatenations of the non-convex constraints from Problem $\mathcal{P}$, and similarly for $\bar{h}(z,\bar{z})$ and $\bar{f}(z,\bar{z})$. The functions in (6) can be thought of as Lagrangian-like functions for the original problem and the convex subproblem, respectively.

The first performance metric is simply the *relative error*

$$\rho = \frac{J(z^*) - L(z^*)}{L(z^*)}, \tag{7}$$

which measures the normalized difference between the functions in (6). The ideal value of the relative error is zero. A second performance metric is the *relative decrease*

$$\rho = \frac{J(\bar{z}) - J(z^*)}{J(\bar{z}) - L(z^*)}, \tag{8}$$

which measures how well the convex subproblem predicts the change in (6a) at the new iterate. The ideal value of the relative decrease is one. Once chosen, the scalar-valued performance metric ρ is used to grow, shrink, or maintain the radius of the trust region. The parameter ρ is also used to reject an iterate z^* in certain cases. If ρ indicates poor prediction, the same convex approximation is kept (by rejecting z^*) but the trust region is shrunk, and the convex subproblem is re-solved. Note that we can assume, without loss of generality, that $\phi(z) > 0$ for any z. Hence (7) and (8) can also be represented using absolute values in the numerator and denominator without affecting the discussion.

In order to decide which case we are in for a given iteration (accept/reject, shrink/keep/grow), the user defines three real numbers $\rho_0,\, \rho_1,\, \rho_2 \in (0,1)$ that split the real number line into four segments. For each iterate, the value of ρ lies in one of these four segments, and the trust region and reference solution are updated according to the rules in Figure 2. The constants $\alpha, \beta > 1$ and $\eta_l > 0$ are user-selected shrink/growth rates and lower bound on the trust region radius. Definition 2.1 defines the class of SCP algorithms studied herein.

Definition 2.1: An algorithm for solving Problem $\mathcal{P}$ belongs to the class $\mathcal{A}_{\mathcal{C}}$ if it iteratively solves Problem $\mathcal{C}$ and uses the performance metric (7) or (8) in conjunction with the appropriate update rules in Figure 2. Moreover, to denote the dependence of this class of algorithms on a set of parameters Π, we write $\mathcal{A}_{\mathcal{C}}(\Pi)$.

Note that in our setup, the set Π in Definition 2.1 is composed of parameters such as ρ_0, ρ_1, ρ_2, α, β, η_l, P, λ and q, but not the initial reference and trust region size.

As a brief technical aside, the denominators in (7) and (8) must be verified to be non-zero before computing the value of ρ. In the case of (7), the condition $\phi(z) > 0$ for all feasible z is sufficient. In the case of (8), a zero denominator actually implies that the iterations can be terminated, since $L(z^*) \le L(\bar{z}) = J(\bar{z})$ (note that this implies the denominator is always nonnegative). If $J(\bar{z}) - L(z^*) = 0$, then the optimal solution found by solving Problem $\mathcal{C}$ is exactly the reference solution, and the iterations can be stopped [30].

III. THE CRAWLING PHENOMENON

An algorithm exhibits the *crawling phenomenon* when it is *forced* to take incrementally smaller steps, as measured by $\|z^* - \bar{z}\|_q$, for an iterate z^* that is not close to a stationary point of the original problem. This section explores the relationship between the crawling phenomenon and the class of SCP algorithms $\mathcal{A}_{\mathcal{C}}$.

A. The Relative Error Case

Consider the sub-class of algorithms $\mathcal{A}^e_{\mathcal{C}} \subset \mathcal{A}_{\mathcal{C}}$ that use the relative error (7) as the performance metric. From Figure 2, any algorithm of this type will shrink the trust region if $\rho \ge \rho_1$, and reject the iterate if $\rho \ge \rho_2$. Hence the trust region is shrunk if

$$\frac{J(z^*) - L(z^*)}{L(z^*)} \ge \rho_1 \quad \Rightarrow \quad J(z^*) \ge (1+\rho_1)L(z^*). \tag{9}$$

Using (6) this can be rewritten as

$$P\big(h(z^*), f(z^*)\big) \ge \frac{\rho_1 \phi(z^*)}{\lambda} + (1+\rho_1) P\big(\bar{h}(z^*,\bar{z}), \bar{f}(z^*,\bar{z})\big). \tag{10}$$

If inequality (10) is satisfied, we find ourselves in the two top-right cases in Figure 2: z^* is either accepted or rejected, and the trust region is shrunk. A sufficient condition for the iterate to be rejected is $P\big(h(z^*), f(z^*)\big) \ge \frac{\phi(z^*)}{\lambda} + 2P\big(\bar{h}(z^*,\bar{z}), \bar{f}(z^*,\bar{z})\big)$, which states that the distance to the feasible set of Problem $\mathcal{P}$ for all iterates rejected by an algorithm in $\mathcal{A}^e_{\mathcal{C}}$ is at least the sum of the (weighted) original cost

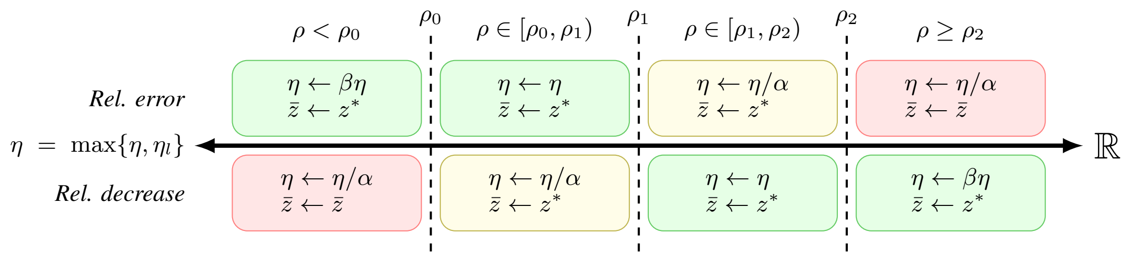

$\rho < \rho_0$
ρ_0
$\rho \in [\rho_0, \rho_1)$
ρ_1
$\rho \in [\rho_1, \rho_2)$
ρ_2
$\rho \geq \rho_2$
Rel. error
$\eta \leftarrow \beta\eta$ $\bar{z} \leftarrow z^*$
$\eta \leftarrow \eta$ $\bar{z} \leftarrow z^*$
$\eta \leftarrow \eta/\alpha$ $\bar{z} \leftarrow z^*$
$\eta \leftarrow \eta/\alpha$ $\bar{z} \leftarrow \bar{z}$
$\eta = \max\{\eta, \eta_l\}$
$\mathbb{R}$
Rel. decrease
$\eta \leftarrow \eta/\alpha$ $\bar{z} \leftarrow \bar{z}$
$\eta \leftarrow \eta/\alpha$ $\bar{z} \leftarrow z^*$
$\eta \leftarrow \eta$ $\bar{z} \leftarrow z^*$
$\eta \leftarrow \beta\eta$ $\bar{z} \leftarrow z^*$

Fig. 2: Update rules for two different performance metrics of a general sequential convex programming algorithm.

and twice the distance to the feasible set of Problem 3. Stated another way, the distance to the feasible set of Problem $\mathcal{P}$ for all iterates *accepted* by the algorithm is bounded above by the same quantity.

Suppose now that the reference $\bar{z}$ is feasible for Problem $\mathcal{P}$, in the sense that $P\big(h(\bar{z}), f(\bar{z})\big) = 0$. This implies that $\bar{z}$ is a feasible point for Problem 3 as well. Thus the optimal solution of Problem $\mathcal{C}$ will not use any virtual control provided that λ is large enough.[1] The solution z^* will therefore satisfy $\bar{h}(z^*, \bar{z}) \leq 0$ and $\bar{f}(z^*, \bar{z}) = 0$. If $\bar{h}(\cdot, \bar{z})$ (resp. $\bar{f}(\cdot, \bar{z})$) does not adequately capture the behaviour of $h(\cdot)$ (resp. $f(\cdot)$), then inequality (10) will be satisfied. This implies that for each instance of Problem $\mathcal{P}$, for every feasible $\bar{z}$ there is an *effective maximum trust region size* that is implicitly defined by λ, the method used to "convexify" the original problem and the value of the original cost. This effective trust region can be quite small when, for example, $f(z)$ has a large Lipschitz constant in the set defined by the trust region. Figure 3 provides a depiction of this scenario; and this discussion leads to the following result.

Lemma 3.1: Given an instance of Problem $\mathcal{P}$ and a reference solution $\bar{z}$ that is feasible for Problem $\mathcal{P}$, the iterate z^* of any algorithm in the sub-class $\mathcal{A}^e_{\mathcal{C}}(\Pi)$ will be used to shrink the trust region if

$$z^* \in \left\{ z \;\middle|\; P\big(h(z), f(z)\big) \geq \tfrac{\rho_1 \phi(z)}{\lambda} \right\}. \tag{11}$$

B. The Relative Decrease Case

Consider now the sub-class of algorithms $\mathcal{A}^d_{\mathcal{C}} \subset \mathcal{A}_{\mathcal{C}}$ that use the relative decrease (8) as the performance metric. In this case, the trust region is shrunk if $\rho < \rho_1$ and rejected when $\rho < \rho_0$. Using (6), the case $\rho < \rho_1$ can be rewritten as,

$$\begin{aligned} P\big(h(z^*), f(z^*)\big) &> \tfrac{1-\rho_1}{\lambda}\left(\phi(\bar{z}) - \phi(z^*)\right) \\ &+ (1-\rho_1) P\big(h(\bar{z}), f(\bar{z})\big) + \rho_1 P\big(\bar{h}(z^*, \bar{z}), \bar{f}(z^*, \bar{z})\big). \end{aligned} \tag{12}$$

If inequality (12) is satisfied, we find ourselves in the two bottom-left cases in Figure 2: the iterate z^* is either accepted or rejected, and the trust region is shrunk. A sufficient condition for the iterate to be rejected is

$$P\big(h(z^*), f(z^*)\big) > \tfrac{\phi(\bar{z}) - \phi(z^*)}{\lambda} + P\big(h(\bar{z}), f(\bar{z})\big). \tag{13}$$

[1]There exists a non-empty feasible set that uses no virtual control, solutions that are not in this set necessarily increase the cost for large values of λ.

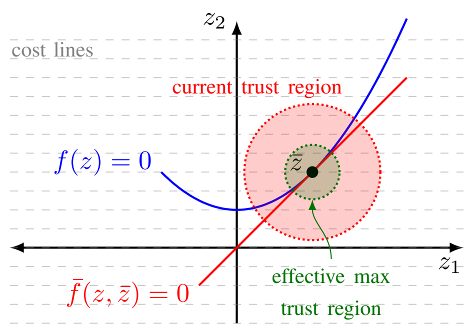


Fig. 3: The effective maximum trust region size for $q = 2$ that induces the crawling phenomenon. The green circle is the largest trust region such that (11) or (14) is *not* satisfied at the resulting iterate. The red trust region would be shrunk until it is a subset of the green trust region.

The inequality (13) states that the distance to the feasible set of Problem $\mathcal{P}$ for any *accepted* iterate is bounded above by the (weighted) decrease in original cost plus the distance between $\bar{z}$ and the feasible set of Problem $\mathcal{P}$.

Suppose now that $\bar{z}$ is feasible for the original problem. By the same logic described in §III-A, we know that the solution z^* will satisfy $\bar{h}(z^*, \bar{z}) \leq 0$ and $\bar{f}(z^*, \bar{z}) = 0$. In this case the final two terms in the right hand side of (12) are zero, as is the second term on the right hand side of (13). We are thus led to the existence of a maximum effective trust region size that is implicitly defined by λ, the method used to "convexify" the original problem and the largest possible reduction in the original cost within the feasible set of Problem 3.

Remark 3.2: The inequality (13) also reveals that if the reference $\bar{z}$ is feasible for Problem $\mathcal{P}$, then any accepted iterate must (strictly) decrease the original cost. A similar fact was noted for SLP in [17], and (13) shows that it holds for the more general class of algorithms $\mathcal{A}^d_{\mathcal{C}}$.

Lemma 3.3: Given an instance of Problem $\mathcal{P}$ and a reference solution $\bar{z}$ that is feasible for Problem $\mathcal{P}$, the iterate z^* of any algorithm in the sub-class $\mathcal{A}^d_{\mathcal{C}}(\Pi)$ will be used to shrink the trust region if

$$z^* \in \left\{ z \;\middle|\; P\big(h(z), f(z)\big) > \tfrac{1-\rho_1}{\lambda}\left(\phi(\bar{z}) - \phi(z)\right) \right\}. \tag{14}$$

The main result of this discussion is summarized in Theorem 3.4. We note that Theorem 3.4 states that the crawling phenomenon *may* be observed, while Lemmas 3.1 and 3.3 indicate *when* it is observed.

Theorem 3.4: Given an instance of Problem $\mathcal{P}$, any algorithm in the class $\mathcal{A}_{\mathcal{C}}(\Pi)$ can exhibit the crawling phenomenon.

C. A Simple Example

This section provides a simple example that exhibits the crawling phenomenon. Consider the following non-convex optimization problem in the variable $z = (z_1, z_2) \in \mathbb{R}^2$,

$$\min_{-2\le z\le 2} \quad z_1 + z_2 \tag{15a}$$

$$\text{s.t.} \quad g_1(z) = -z_2 - \tfrac{4}{3}z_1 - \tfrac{2}{3} \le 0 \tag{15b}$$

$$f(z) = z_2 - z_1^4 - 2z_1^3 + 1.2z_1^2 + 2z_1 = 0 \tag{15c}$$

Following the procedure outlined in §II, we can convexify Problem 15 in the form of Problem $\mathcal{C}$ by using a reference $\bar{z} \in \mathbb{R}^2$ and virtual control $\nu \in \mathbb{R}$. The parameters used for this particular example are an initial trust region of $\eta = 0.1$, an initial reference of $\bar{z} = (1.5, 1.5)$, a penalty function $P(\nu) = |\nu|$ and $\lambda = 400$. The values used to update the trust region size are $(\rho_0, \rho_1, \rho_2) = (0, 0.1, 0.9)$ and $(\rho_0, \rho_1, \rho_2) = (0.1, 0.9, 1.0)$ for the relative decrease and relative error respectively, and $\alpha = 1.5$, $\beta = 2.0$, $q = 2$. Figure 4 shows the resulting sequence of iterates for both performance metrics, as well as the value of ρ versus the iteration number. Figures 4a and 4b demonstrate the crawling phenomenon; progress towards a solution slows down considerably once the curvature of the equality constraint increases towards the bottom, and Figure 4c confirms that this is a result of a shrinking trust region. In both cases, Figure 4c reveals a cycle that is reached whereby the approximation accuracy slowly worsens until the trust region is shrunk (the sharp changes in ρ). This is a result of the nonlinear equality constraint bending away from the affine approximation, eventually leading to (11) or (14).

It is important to resist the temptation to draw overly general conclusions from this simple example. For other choices of initial reference $\bar{z}$, the crawling phenomenon is avoided. For example, convergence to the local minimum from an initial reference in the (infeasible) shaded gray region is quite rapid, often terminating in ten or fewer iterations. There are, however, distinct regions in the upper-middle and upper-right of the area plotted in Figure 4 from which the phenomenon is observed. This is inline with Theorem 3.4, which states only that the crawling phenomenon *can* happen. Moreover, it is clear that for this example the crawling phenomenon is occurring. This is only obvious because we know exactly where each minimum is (i.e., we can plot the cost lines, constraints and iterates). The trouble, in general, is that it is not possible to know (or plot) the solutions to Problem $\mathcal{P}$. Algorithms in the class $\mathcal{A}_{\mathcal{C}}$ are designed to find these solutions, often for the first time, and so it may be unknown whether slow progress means one is close to a stationary point, or if it is the crawling phenomenon.

IV. POSSIBLE REMEDIES

The following scheme can be used to identify if the crawling phenomenon is occurring. If the algorithm at iteration $k > 3$: (i) chooses to shrink the trust region, (ii) accepted iterates $k-1, k-2, k-3$, (iii) $\rho \in [\rho_{lb}, \rho_{ub}]$ for iterates $k-1, k-2, k-3$ and (iv) cases (i)-(iii) have been met twice, then the crawling phenomenon is occurring. The interval $[\rho_{lb}, \rho_{ub}]$ is a subset of $[\rho_1, \rho_2]$ for the relative error and of $[\rho_0, \rho_1]$ for the relative decrease. Based on Theorem 3.4, we must turn to algorithms that do not belong to the class $\mathcal{A}_{\mathcal{C}}$ if we are to avoid the crawling phenomenon once it has been detected. In general, algorithms that belong to $\mathcal{A}_{\mathcal{C}}(\Pi)$ are quite good at quickly reaching feasibility for Problem $\mathcal{P}$[2]. Hence we may propose hybrid algorithms that are formed by combining one algorithm in the class $\mathcal{A}_{\mathcal{C}}$ with a second that is not, but which assumes a feasible initial guess for the original problem. To illustrate this capability, when the crawling phenomenon is detected in the example from §III-C, the algorithm is switched to the feasible SQP method described in [33]. This algorithm guarantees that all subsequent iterates are feasible for Problem $\mathcal{P}$, but is not of trust region type, and does not convexify the problem in the same way as in Problem $\mathcal{C}$, hence does not belong to the class $\mathcal{A}_{\mathcal{C}}$. The resulting iterates of this hybrid algorithm are shown in Figure 5. One can see that the crawling phenomenon, once detected, is avoided, and the overall number of iterations is greatly reduced. This hybrid algorithm effectively blends the advantages of both algorithms.

V. CONCLUSIONS

This paper has shown that all SCP algorithms that belong to the class $\mathcal{A}_{\mathcal{C}}$ are susceptible to the crawling phenomenon. We have given an algebraic description of when two subclasses of $\mathcal{A}_{\mathcal{C}}$ will exhibit the behaviour, and offered geometric interpretations. As SCP algorithms become increasingly popular, it is important to recognize their limitations. At the same time, knowledge of said limitations provides an opportunity to design better algorithms if needed. The use of hybrid algorithms was shown to be a promising candidate to mitigate the effect of the crawling phenomenon in sequential convex programming.

ACKNOWLEDGMENT

The authors thank Miki Szmuk, Danylo Malyuta and Behçet Açıkmeşe for many valuable discussions on the subject of sequential convex programming.

REFERENCES

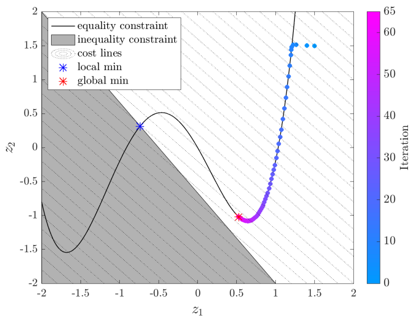
[1] B. Açıkmeşe and L. Blackmore, “Lossless Convexification of a Class of Optimal Control Problems with Non-Convex Control Constraints,” *Automatica*, vol. 47, no. 2, pp. 341–347, 2011.

[2] L. Blackmore, B. Açıkmeşe, and J. M. Carson III, “Lossless Convexification of Control Constraints for a Class of Nonlinear Optimal Control Problems,” *Systems and Control Letters*, vol. 61, no. 8, pp. 863–870, 2012.

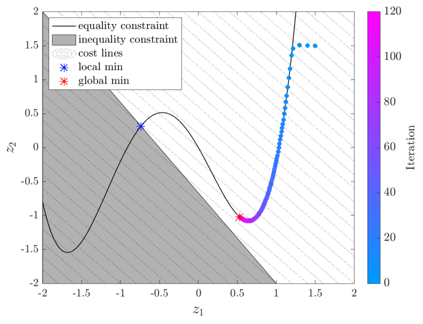
[3] M. W. Harris and B. Açıkmeşe, “Lossless Convexification of Non-Convex Optimal Control Problems for State Constrained Linear Systems,” *Automatica*, vol. 50, no. 9, pp. 2304–2311, 2014.

[4] M. Szmuk, T. P. Reynolds, and B. Açıkmeşe, “Successive Convexification for Real-Time 6-DoF Powered Descent Guidance with State-Triggered Constraints,” *arXiv e-prints*, 2018. arXiv:1811.10803.

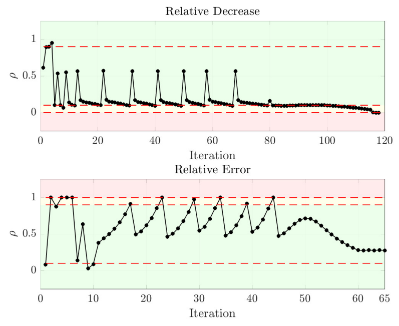
[2]Prior to feasibility, the solutions of Problem $\mathcal{C}$ are the closest point in the trust region to the projection of $\bar{z}$ onto the feasible set of Problem 3. In the relative decrease case, in fact, one can show that $J(\bar{z}) > J(z^*)$.



(a) Relative error solution.

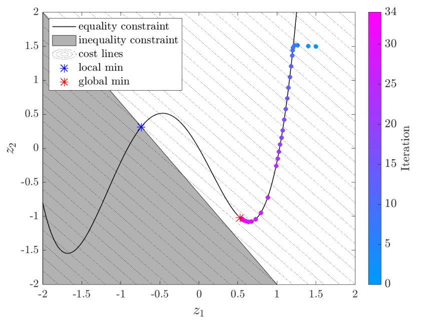


(b) Relative decrease solution.

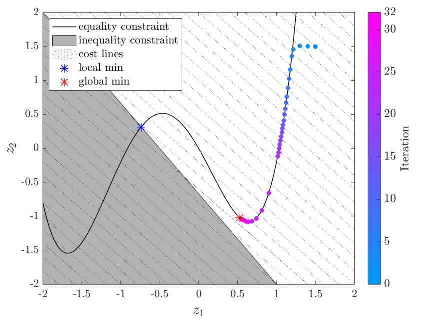


(c) The performance metric ρ saturated to the $[0, 1]$ interval solely for plotting. Coloured regions correspond to Figure 2.

Fig. 4: Iterates of two algorithms from $\mathcal{A}_\mathcal{C}$ that show the crawling phenomenon for Problem 15.



(a) Relative error solution.



(b) Relative decrease solution.

Fig. 5: Iterates of two hybrid algorithms that avoid the crawling phenomenon for Problem 15.

[5] U. Lee and M. Mesbahi, "Constrained Autonomous Precision Landing via Dual Quaternions and Model Predictive Control," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 292–308, 2017.

[6] X. Liu and P. Lu, "Solving Nonconvex Optimal Control Problems by Convex Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 750–765, 2014.

[7] A. Beck, A. Ben-Tal, and L. Tetruashvili, "A Sequential Parametric Convex Approximation Method with Applications to Nonconvex Truss Topology Design Problems," *Journal of Global Optimization*, vol. 47, pp. 29–51, 2010.

[8] W. Wei, J. Wang, N. Li, and S. Mei, "Optimal Power Flow of Radial Networks and Its Variations: A Sequential Convex Optimization Approach," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2974–2987, 2017.

[9] H. Jiang, M. S. Drew, and Z.-N. Li, "Matching by Linear Programming and Successive Convexification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 959–975, 2007.

[10] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. Academic Press, Inc., 1981.

[11] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY: Springer Science & Business Media, 2006.

[12] P. A. Parrilo, "Semidefinite Programming Relaxations for Semialgebraic Problems," *Math. Program., Ser. B*, vol. 96, pp. 293–320, 2003.

[13] G. Blekherman, P. A. Parrilo, and R. R. Thomas, eds., *Semidefinite Optimization and Convex Algebraic Geometry*. SIAM, 2012.

[14] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[16] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz, "An Algorithm for Nonlinear Optimization Using Linear Programming and Equality Constrained Subproblems," *Math. Program., Ser. B*, vol. 100, pp. 27–48, 2004.

[17] F. Palacios-Gomez, L. Lasdon, and M. Engquist, "Nonlinear Optimization by Successive Linear Programming," *Management Science*, vol. 28, no. 10, pp. 1106–1120, 1982.

[18] R. Horst and N. V. Thoai, "Dc programming: Overview," *Journal of Optimization Theory and Applications*, vol. 103, no. 1, pp. 1–43, 1999.

[19] A. Yuille and A. Rangaranjan, "The Concave-Convex Procedure (CCCP)," in *Advances in Neural Information Processing Systems*, pp. 1033–1040, 2002.

[20] G. R. Lanckriet and B. K. Sriperumbudur, "On the Convergence of the Concave-Convex Procedure," in *Advances in Neural Information Processing Systems 22*, pp. 1759–1767, 2009.

[21] S. P. Han, "A Globally Convergent Method for Nonlinear Programming," *Journal of Optimization Theory and Applications*, vol. 22, no. 3, pp. 297–309, 1977.

[22] M. J. Powell, "Algorithms for Nonlinear Constraints that use Lagrangian Functions," *Mathematical Programming*, vol. 14, no. 1, pp. 224–248, 1978.

[23] M. J. Powell and Y. Yuan, "A Recursive Quadratic Programming Algorithm that uses Differentiable Exact Penalty Functions," *Mathematical Programming*, vol. 35, pp. 265–278, 1986.

[24] P. T. Boggs and W. J. Tolle, "A Strategy for Global Convergence in a Sequential Quadratic Programming Algorithm," *SIAM Journal of Numerical Analysis*, vol. 26, no. 3, pp. 600–623, 1989.

[25] P. T. Boggs and W. J. Tolle, "Sequential Quadratic Programming," *Acta Numerica*, vol. 4, pp. 1–52, 1996.

[26] M. Fukushima, "A Successive Quadratic Programming Algorithm with Global and Superlinear Convergence Properties," *Mathematical Programming*, vol. 35, pp. 253–264, 1986.

[27] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *Siam Review*, vol. 47, no. 1, pp. 99–131, 2005.

[28] J. T. Betts and W. P. Huffman, "Path-Constrained Trajectory Optimization using Sparse Sequential Quadratic Programming," *Journal of Guidance, Control, and Dynamics*, vol. 16, pp. 59–68, Jan. 1993.

[29] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive Convexification of Non-Convex Optimal Control Problems and its Convergence Properties," in *IEEE Conference on Decision and Control*, (Las Vegas, NV), 2016.

[30] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems," *arXiv e-prints*, 2018. arXiv:1804.06539.

[31] R. Bonalli, A. Bylard, A. Cauligi, T. Lew, and M. Pavone, "Trajectory Optimization on Manifolds: A Theoretically-Guaranteed Embedded Sequential Convex Programming Approach," in *Robotics: Science and Systems*, June 2019.

[32] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[33] C. T. Lawrence and A. L. Tits, "A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm," *SIAM Journal on Optimization*, vol. 11, no. 4, pp. 1092–1118, 2003.