2020 European Control Conference (ECC)
May 12-15, 2020. Saint Petersburg, Russia

Determining the Exact Local Convergence Rate of Sequential Convex Programming

Florian Messerer* and Moritz Diehl**

***Abstract*— Sequential Convex Programming (SCP) is an iterative algorithm for solving Nonlinear Programs (NLP) with "convex-over-nonlinear" substructure. At every iteration it solves a convex, but generally nonlinear, approximation to the original NLP, exploiting its outer convexities. It is already known that SCP has linear local convergence, though without a tight characterization of the rate. Sequential Convex Quadratic Programming (SCQP) is another "convex-over-nonlinear" substructure exploiting algorithm, for which a tight characterization of the convergence rate has already been obtained. In this paper we show under mild assumptions that in fact both methods have the same local linear convergence – or divergence – rate. We can therefore determine the convergence rate of SCP, which is tightly characterized by two simple matrix inequalities evaluated at the solution. We further reason why – as a heuristic – SCP should in general show more robust global convergence than SCQP. We illustrate this, as well as the local convergence rate, with a numerical example.**

I. Introduction

This paper compares two algorithms for the solution of constrained nonlinear programs (NLP) with "convex-over-nonlinear" substructure

$$\min_{w \in \mathbb{R}^n} \quad \phi_0(F_0(w)) \tag{1a}$$
$$\text{s.t.} \quad \phi_i(F_i(w)) \leq 0, \qquad i = 1, \dots, q, \tag{1b}$$
$$g(w) = 0 \tag{1c}$$

with outer convexities $\phi_i : \mathbb{R}^m \to \mathbb{R}$, inner nonlinearities $F_i : \mathbb{R}^n \to \mathbb{R}^m$ and nonlinear equality constraints $g : \mathbb{R}^n \to \mathbb{R}^p$. We assume that the F_i and g are twice, the ϕ_i thrice continuously differentiable. For ease of notation, intermediate dimension m is assumed to be identical for all i, but the results of this paper hold for individualized m_i all the same. We further define $f_i(w) := \phi_i(F_i(w))$. Note that every (twice continuously differentiable) nonlinear function $f_i(\cdot)$ can be brought into this form, as the corresponding $\phi_i(\cdot)$ could always be, e.g., the identity mapping.

Both algorithms iterate by solving convex approximations to (1), which exploit the convex over nonlinear substructures. But they differ in the way they construct these approximations. Sequential Convex Programming (SCP) keeps the outer convexities as they are, and only linearizes the inner nonlinearities. In consequence it solves a convex – but generally nonlinear problem – at every iteration. Sequential Convex Quadratic Programming (SCQP) on the other hand approximates the original NLP by a convex quadratic program (QP), thus linearizing all constraints, but keeps contributions of the outer convexities as part of its Hessian. Both methods can be seen as generalizations of Constrained Gauss-Newton (CGN) [2] into different directions. For the case of a nonlinear least squares objective, and if the outer convexities of the constraints are taken to be the identity mapping, both methods are equivalent to CGN. Furthermore, in the unconstrained case SCQP is the same as Generalized Gauss-Newton[1] [9].

A. Contribution and related work

SCP has been introduced in [10], where locally linear convergence of this method has been proven, but without a tight characterization of the contraction rate. SCQP originates from [11], in which not only linear convergence of this has been proven, but a tight characterization of this contraction rate already has already been obtained. We build on these results and compare the two algorithms by developing in parallel an analysis of their convergence behaviour. As the main result we then show that SCP and SCQP actually have identical asymptotic local linear contraction – or divergence – rate. A similar result for the unconstrained case and the methods SCP and GGN has already been obtained in [3]. We then explain why we expect SCP to show more robust global convergence behavior and demonstrate this with a numerical example.

B. Notation and Preliminaries

We denote by $\frac{\partial f}{\partial w}(w) \in \mathbb{R}^{m \times n}$ the Jacobian of a function $f : \mathbb{R}^n \to \mathbb{R}^m$, $w \mapsto f(w)$, and the gradient is its transpose, $\nabla f(w) = \frac{\partial f}{\partial w}(w)^\top$. Gradient operator ∇ always refers to the first argument of a function, unless the argument is explicitly written as subscript. For the Jacobian of $F_i(w)$ we use the shortcut $J_i(w)$. If $f(\cdot)$ is scalar valued, its Hessian is denoted by $\nabla^2 f(w)$. Linearizations are referred to as

$$f^{\text{lin}}(w; \bar{w}) := f(\bar{w}) + \nabla f(\bar{w})^\top (w - \bar{w}). \tag{2}$$

The notation $f_{\mathcal{S}}(w)$, for a set of indices $\mathcal{S} \subset \{0, \dots, q\}$, means the vertical concatenation of the $f_i(w)$, $i \in \mathcal{S}$. Similarly, for a vector $\mu \in \mathbb{R}^{q+1}$, $\mu_{\mathcal{S}}$ is the vector slice of the corresponding indices. Slightly less conventional,

This research was supported by DFG via Research Unit FOR 2401.

*Department of Microsystems Engineering (IMTEK), University of Freiburg, 79110 Freiburg, Germany florian.messerer@imtek.de

**Department of Microsystems Engineering (IMTEK) and Department of Mathematics, University of Freiburg, 79110 Freiburg, Germany moritz.diehl@imtek.de

[1]The method we call Constrained Gauss-Newton in this paper is originally referred to as Generalized Gauss-Newton (GGN). We follow the terminology from the field of computer science and reserve the name 'GGN' for the method introduced in [9].

$\phi_{\mathcal{S}}(F_{\mathcal{S}}(w))$ denotes the concatenation of the corresponding $\phi_i(F_i(w))$. For a more lightweight notation the vertical concatenation $[x^\top, y^\top]^\top$ of two vectors $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, is denoted by (x, y). $\|\cdot\|$ denotes a general vector norm. More specifically, $\|\cdot\|_2$ is the Euclidean norm and $\|\cdot\|_\infty$ the maximum norm. $\|x\|_W^2$ with $W \in \mathbb{R}^{n\times n}$ means $x^\top W x$.

Concepts from the field of numerical optimization are defined as in [7] unless otherwise stated. We abbreviate by LICQ the linear independence constraint qualification, and KKT refers to the Karush Kuhn Tucker conditions.

II. Sequential Convex Programming

Sequential Convex Programming (SCP) solves NLP (1) by iteratively solving its convex approximation, which is obtained through linearization of inner nonlinearities F_i and equality constraints g, while keeping the outer convexities ϕ_i. Starting at initial guess w_0 the method iterates as

$$w_{k+1} \in \underset{w \in \mathbb{R}^n}{\arg\min} \quad \phi_0(F_0^{\text{lin}}(w; w_k)) \tag{3a}$$

$$\text{s.t.} \quad \phi_i(F_i^{\text{lin}}(w; w_k)) \leq 0, \quad i = 1, \ldots, q, \tag{3b}$$

$$g^{\text{lin}}(w; w_k) = 0 \tag{3c}$$

with linearizations as defined in (2). We introduce the shorthand $f_i^{\text{SCP}}(w; \bar{w}) := \phi_i(F_i^{\text{lin}}(w; \bar{w}))$. Problem (3) also defines Lagrange multipliers $\mu_{k+1} \in \mathbb{R}^q$ and $\lambda_{k+1} \in \mathbb{R}^p$ corresponding to primal solution w_{k+1}. If we collect these in $z_{k+1} = (w_{k+1}, \mu_{k+1}, \lambda_{k+1})$ and furthermore assure that z_{k+1} is uniquely defined, e.g., by choosing the minimum norm solution, then (3) defines the iteration map

$$z_{k+1} = z_{\text{SCP}}^{\text{sol}}(z_k). \tag{4}$$

Note that though the full z_k is taken as input here, the output actually only depends on w_k. Under mild assumptions SCP has locally linear convergence, but so far the rate has not been tightly characterized [10].

Let us now state a basic property on stationarity of SCP.

Lemma 1: Denote by $\mathcal{Z}^*$ the set of KKT points of (1). If $\bar{z}$ is a fixed point of (4) at which LICQ holds, then $\bar{z} \in \mathcal{Z}^*$. Vice versa, if $z^* \in \mathcal{Z}^*$, and the solution to (3) is unique at z^*, then z^* is a fixed point of (4).

Proof: Assume $\bar{z}$ is a fixed point of (4). This means that $\bar{z}$ solves (3) and, due to LICQ, is also a KKT point. Then, by substituting $z_{k+1} = z_k = \bar{z}$ into the KKT conditions of (3) – which we know to hold in this case – they collapse to the KKT conditions of NLP (1) – which therefore also hold. It follows that $\bar{z} \in \mathcal{Z}^*$. Vice versa, assume that $z^* \in \mathcal{Z}^*$. Writing down the KKT conditions of (3), we can see that they hold at $z_{k+1} = z_k = z^*$. Therefore, due to convexity of (3), z^* is a solution of (3) at z^* and, due to uniqueness, stationary in (4). ∎

A direct consequence of Lemma 1, and the fact that $f_i^{\text{SCP}}(w^*; w^*) = f_i(w^*)$, is the following Corollary.

Corollary 1: Strict complementarity of the KKT conditions of (1) holds at $z^* \in \mathcal{Z}^*$, if and only if it also holds for (3) at z^*. More specifically, the corresponding multipliers – and therefore the active set – are identical.

Before stating the next Lemma, let us first introduce some notation. $B_{\text{SCP}}(z; \bar{z})$ denotes the Hessian of the Lagrangian of (3), and $B_* := B_{\text{SCP}}(z^*; z^*)$. $\mathcal{A}$ is the active set of (1) at $z^* \in \mathcal{Z}^*$, with $\mu_{\mathcal{A}}$ the corresponding multipliers. Analogously we have $\mathcal{I}$ and $\mu_{\mathcal{I}}$ for the inactive constraints. All active constraints, including the equalities, are collected in

$$G_{\text{SCP}}(w; \bar{w}) := \begin{bmatrix} \phi_{\mathcal{A}}(F_{\mathcal{A}}^{\text{lin}}(w; \bar{w})) \\ g^{\text{lin}}(w; \bar{w}) \end{bmatrix} \tag{5}$$

with corresponding multipliers $\gamma = (\mu_{\mathcal{A}}, \lambda)$. Finally,

$$\Gamma_* := \frac{\partial G_{\text{SCP}}}{\partial w}(w^*; w^*) = [\nabla f_{\mathcal{A}}(w^*)^\top, \nabla g(w^*)^\top]. \tag{6}$$

The following Lemma is actually a direct consequence of [8], which proves it for a more general class of methods. Nonetheless a proof specific to SCP will be stated here, to introduce the reader to some ideas which will be central to the later results.

Lemma 2: Assume LICQ and strict complementarity hold at $z^* \in \mathcal{Z}^*$, and that $Z^\top B_* Z \succ 0$, with Z a basis of the nullspace of Γ_*. Then, for $\bar{z}$ sufficiently close to z^*, the active set of problem (3) is stable and corresponds to $\mathcal{A}$, the active set of original NLP (1) at z^*.

Proof: From Corollary 1 we know that at z^* the active sets of (1) and (3) are identical. Now consider the residuals

$$R_{\text{SCP}}(y; \bar{y}) := \begin{bmatrix} \nabla f_0^{\text{SCP}}(w; \bar{w}) + \nabla G_{\text{SCP}}(w; \bar{w})\gamma \\ G_{\text{SCP}}(w; \bar{w}) \end{bmatrix} \tag{7}$$

with $y = (w, \gamma)$. We have $R_{\text{SCP}}(y^*; y^*) = 0$, and $R_{\text{SCP}}(\cdot;\cdot)$ is continuously differentiable with respect to both arguments, due to the assumptions made on the composing functions. Furthermore,

$$\frac{\partial R_{\text{SCP}}}{\partial y}(y^*; y^*) = \begin{bmatrix} B_* & \Gamma_*^\top \\ \Gamma_* & 0 \end{bmatrix} \tag{8}$$

is invertible due to the KKT Lemma [7, Lem. 16.1], which holds because of LICQ and $Z^\top B_* Z \succ 0$. We thus know from the implicit function theorem that there exists a neighborhood of y^* in which continuously differentiable function $y_{\text{SCP}}^{\text{sol}}(\bar{y})$ is uniquely defined by $R_{\text{SCP}}(y_{\text{SCP}}^{\text{sol}}(\bar{y}); \bar{y}) = 0$. We can decompose it into $y_{\text{SCP}}^{\text{sol}}(\bar{y}) = (w_{\text{SCP}}^{\text{sol}}(\bar{y}), \mu_{\mathcal{A},\text{SCP}}^{\text{sol}}(\bar{y}), \lambda_{\text{SCP}}^{\text{sol}}(\bar{y}))$. Furthermore $y_{\text{SCP}}^{\text{sol}}(y^*) = y^*$. Since $\mu_{\mathcal{A}}^* > 0$ and $f_{\mathcal{I}}^{\text{SCP}}(w^*; w^*) < 0$, due to continuity, and for $\bar{y}$ sufficiently close to y^*, we also have $\mu_{\mathcal{A},\text{SCP}}^{\text{sol}}(\bar{y}) < 0$ and $f_{\mathcal{I}}^{\text{SCP}}(w_{\text{SCP}}^{\text{sol}}(\bar{w}); \bar{w}) < 0$. Therefore in a neighborhood of w^* the active set of (3) is stable. ∎

III. Sequential Convex Quadratic Programming

Consider again our original problem (1). Its Lagrangian is given by

$$\mathcal{L}(w, \lambda, \mu) = f_0(w) + \sum_{i=1}^{q} \mu_i f_i(w) + \sum_{i=1}^{p} \lambda_i g_i(w) \tag{9}$$

with multipliers $\mu \in \mathbb{R}^q$ and $\lambda \in \mathbb{R}^p$. Its Hessian can then be split into two contributions,

$$\nabla^2 \mathcal{L}(w, \lambda, \mu) = B_{\text{SCQP}}(w, \mu) + E_{\text{SCQP}}(w, \lambda, \mu), \tag{10}$$

where the positive semi-definite

$$B_{\mathrm{SCQP}}(w,\mu) := B_0(w) + \sum_{i=1}^{q} \mu_i B_i(w) \qquad \text{with} \tag{11}$$

$$B_i(w) = J_i(w)^\top \nabla^2 \phi_i(F_i(w)) J_i(w), \quad i = 0,\dots,q \tag{12}$$

is the Sequential Convex Quadratic Programming (SCQP) Hessian approximation [11]. The corresponding approximation error is

$$E_{\mathrm{SCQP}}(w,\lambda,\mu) := \sum_{j=1}^{m} \nabla^2 F_{0,j}(w) \frac{\partial \phi_0}{\partial F_{0,j}}(F_0(w)) + \sum_{i=1}^{q} \mu_i \sum_{j=1}^{m} \nabla^2 F_{i,j}(w) \frac{\partial \phi_i}{\partial F_{i,j}}(F_i(w)) + \sum_{i=1}^{p} \lambda_i \nabla^2 g_i(w), \tag{13}$$

where $F_{i,j}$ denotes the j-th entry of F_i. Now SCQP solves at every iteration a convex QP approximation

$$\min_{w \in \mathbb{R}^n} \quad f_0(w_k) + \nabla f_0(w_k)^\top (w - w_k) + \tfrac{1}{2}(w - w_k)^\top B_{\mathrm{SCQP}}(w_k,\mu_k)(w - w_k) \tag{14a}$$

$$\text{s.t.} \quad f_i^{\mathrm{lin}}(w; w_k) \le 0, \quad i = 1,\dots,q, \tag{14b}$$

$$g^{\mathrm{lin}}(w; w_k) = 0 \tag{14c}$$

of the original problem (1). If w_{k+1} is a solution to (14) with corresponding multipliers μ_{k+1} and λ_{k+1}, collected in $z_{k+1} = (w_{k+1}, \mu_{k+1}, \lambda_{k+1})$, then (14) defines the iteration

$$z_{k+1} = z^{\mathrm{sol}}_{\mathrm{SCQP}}(z_k). \tag{15}$$

As for SCP before, we have assumed that some measure has been taken such that z_{k+1} is uniquely defined. Note that λ_k has been included in the argument of (15), but the iteration actually only depends on w_k and μ_k. In [11] its linear local convergence has been proven under mild assumptions while also giving a tight characterization of the rate.

Let us now state two standard results of Sequential Quadratic Programming without proof. The interested reader is referred to [8] or can follow the proofs of Lemmas 1 and 2, as they are analogous in structure.

Lemma 3: Denote by $\mathcal{Z}^*$ the set of KKT points of (1). If $\bar{z}$ is a fixed point of (15) at which LICQ holds, then $\bar{z} \in \mathcal{Z}^*$. Vice versa, if $z^* \in \mathcal{Z}^*$, and the solution to (14) is unique at z^*, then z^* is a fixed point of (15).

Lemma 4: Assume LICQ and strict complementarity hold at $z^* \in \mathcal{Z}^*$, and that $Z^\top B_* Z \succ 0$, with Z a basis of the nullspace of Γ_*. Then, for $\bar{z}$ sufficiently close to z^*, the active set of problem (14) is stable and corresponds to $\mathcal{A}$, the active set of original NLP (1) at z^*.

Finally, for later use let us define the SCQP residual map analogously to (7):

$$R_{\mathrm{SCQP}}(y;\bar{y}) := \begin{bmatrix} \nabla f_0(\bar{w}) + B_{\mathrm{SCQP}}(\bar{w},\bar{\mu})(w - \bar{w}) + \Gamma(\bar{w})^\top \gamma \\ G_{\mathrm{SCQP}}(w;\bar{w}) \end{bmatrix}, \tag{16}$$

where $G_{\mathrm{SCQP}}(w;\bar{w}) := (f_{\mathcal{A}}^{\mathrm{lin}}(w;\bar{w}), g^{\mathrm{lin}}(w;\bar{w}))$ and $\Gamma(\bar{w}) := \frac{\partial G_{\mathrm{SCQP}}}{\partial w}(w;\bar{w}) = [\nabla f_{\mathcal{A}}(\bar{w})^\top, \nabla g(\bar{w})^\top]$.

IV. Local Convergence Analysis

Before deriving the convergence rate of both SCP and SCQP we will introduce the following Lemma, which characterizes the similarity of the two methods and is at the core of the main theorem of this paper.

Lemma 5: Consider the residual maps $R_{\mathrm{SCP}}(y;\bar{y})$ and $R_{\mathrm{SCQP}}(y;\bar{y})$ as defined in (7) and (16) respectively. It holds

$$R_{\mathrm{SCP}}(y;\bar{y}) = R_{\mathrm{SCQP}}(y;\bar{y}) + \mathcal{O}(\|y - \bar{y}\|^2). \tag{17}$$

Proof: The proof will go by showing identity of the residuals and their partial derivatives at $y = \bar{y}$. Before we start, note that $f_i^{\mathrm{lin}}(\bar{w};\bar{w}) = f_i(\bar{w})$, and $\nabla f_i(w) = J_i(w)^\top \nabla \phi_i(F_i(w))$. The first part of the proof simply goes by seeing that

$$R_{\mathrm{SCP}}(\bar{y};\bar{y}) = \begin{bmatrix} \bar{J}_0^\top \nabla \phi_0(\bar{F}_0) + \sum_{i\in\mathcal{A}} \bar{\mu}_i \bar{J}_i^\top \nabla \phi_i(\bar{F}_i) + \nabla \bar{g} \bar{\lambda} \\ \phi_{\mathcal{A}}(\bar{F}_{\mathcal{A}}) \\ \bar{g} \end{bmatrix} = \begin{bmatrix} \nabla \bar{f}_0 + \sum_{i\in\mathcal{A}} \bar{\mu}_i \nabla \bar{f}_i + \nabla \bar{g} \bar{\lambda} \\ \bar{f}_{\mathcal{A}} \\ \bar{g} \end{bmatrix} = R_{\mathrm{SCQP}}(\bar{y};\bar{y}),$$

with shorthands $\bar{J}_i := J_i(\bar{w})$, $\bar{F}_i := F_i(\bar{w})$, $\bar{f}_i := f_i(\bar{w})$, $\bar{g} := g(\bar{w})$, $\nabla \bar{g} := \nabla g(\bar{w})$.

For the first order derivatives we have,

$$\frac{\partial R_{\mathrm{SCP}}}{\partial y}(y;\bar{y}) = \begin{bmatrix} B_{\mathrm{SCP}}(w,\mu;\bar{w}) & \nabla f_{\mathcal{A}}^{\mathrm{SCP}}(w;\bar{w}) & \nabla g(\bar{w}) \\ \nabla f_{\mathcal{A}}^{\mathrm{SCP}}(w;\bar{w})^\top & 0 & 0 \\ \nabla g(\bar{w})^\top & 0 & 0 \end{bmatrix} \tag{18}$$

with

$$B_{\mathrm{SCP}}(w,\mu;\bar{w}) := \check{B}_0(w;\bar{w}) + \sum_{i\in\mathcal{A}} \mu_i \check{B}_i(w;\bar{w}) \quad \text{and} \tag{19}$$

$$\check{B}_i(w;\bar{w}) := \bar{J}_i^\top \nabla^2 \phi_i(F_i^{\mathrm{lin}}(w;\bar{w})) \bar{J}_i, \quad i = 0,\dots,q. \tag{20}$$

Evaluating at $\bar{y}$ then leads to

$$\frac{\partial R_{\mathrm{SCP}}}{\partial y}(\bar{y};\bar{y}) = \begin{bmatrix} \bar{B}_0 + \sum_{i\in\mathcal{A}} \bar{\mu}_i \bar{B}_i & \nabla \bar{f}_{\mathcal{A}} & \nabla \bar{g} \\ \nabla \bar{f}_{\mathcal{A}}^\top & 0 & 0 \\ \nabla \bar{g}^\top & 0 & 0 \end{bmatrix} = \begin{bmatrix} \bar{B} & \nabla \bar{f}_{\mathcal{A}} & \nabla \bar{g} \\ \nabla \bar{f}_{\mathcal{A}}^\top & 0 & 0 \\ \nabla \bar{g}^\top & 0 & 0 \end{bmatrix} = \frac{\partial R_{\mathrm{SCQP}}}{\partial y}(\bar{y};\bar{y})$$

with $\bar{B}_i := B_i(\bar{w})$, $B_i(\cdot)$ as defined in (12), and $\bar{B} := B_{\mathrm{SCQP}}(\bar{w},\bar{\mu}) = B_{\mathrm{SCP}}(\bar{w},\bar{\mu};\bar{w})$. ∎

After having shown this connection between SCP and SCQP we can now move to the main theorem of this script, which characterizes the local contraction rate of the two algorithms. For the case of SCQP this result has already been proven in [11] in a similar way. Before we state and prove the theorem, recall that Γ_* is the Jacobian of the extended equality constraints at the solution, defined in (6), and Z

a basis of its nullspace. The reduced Hessian of the Lagrangian is $\widetilde{\Lambda}_* := Z^\top \nabla^2 \mathcal{L}(w^*, \lambda^*, \mu^*) Z$ with corresponding approximation $\widetilde{B}_* := Z^\top B_* Z$ and error $\widetilde{E}_* := Z^\top E_* Z$, where $B_* := B_{\mathrm{SCQP}}(w^*, \mu^*) = B_{\mathrm{SCP}}(w^*, \mu^*; w^*)$ and $E_* = E_{\mathrm{SCQP}}(w^*, \lambda^*, \mu^*)$.

Theorem 1: Assume z^* is a local minimizer of optimization problem (1), with LICQ, strict complementarity, and $\widetilde{B}_* \succ 0$. Then z^* is a fixed point of both SCQP and SCP and both methods have the same asymptotic local linear contraction – or divergence – rate. This asymptotic contraction rate is given by the smallest $\alpha \geq 0$ that still fulfills the conditions

$$-\alpha \widetilde{B}_* \preceq \widetilde{E}_* \preceq \alpha \widetilde{B}_*. \tag{21}$$

Accordingly, if

$$\frac{1}{1+\alpha} \widetilde{\Lambda}_* \preceq \widetilde{B}_* \preceq \frac{1}{1-\alpha} \widetilde{\Lambda}_* \tag{22}$$

holds for some $\alpha < 1$, the methods converge linearly with contraction rate α – or diverges if it only holds for $\alpha > 1$. In consequence, a necessary condition for local convergence is given by $\widetilde{B}_* \succeq \frac{1}{2}\widetilde{\Lambda}_*$. Finally, if $\widetilde{\Lambda}_* \succ 0$, then $\widetilde{B}_* \succ \frac{1}{2}\widetilde{\Lambda}_*$ is sufficient for local linear convergence.

Proof: Stationarity follows from Lemmas 1 and 3. From Lemmas 2 and 4 we know that the active set $\mathcal{A}$ of both methods is stable close to z^* and corresponds to the true one. It follows that the $\mu_\mathcal{I}$ have already converged, $\mu_\mathcal{I} = \mu^*_\mathcal{I} = 0$ for z sufficiently close to z^*. We can therefore disregard the inactive constraints and characterize the convergence behavior from the residuals maps $R_i(y; \bar{y})$ as defined in (7) and (16) for $i = \mathrm{SCP}, \mathrm{SCQP}$. As a consequence of Lemma 5 their partial derivatives are identical at $y = \bar{y} = y^*$ and given by

$$\frac{\partial R_i}{\partial y}(y^*; y^*) = \begin{bmatrix} B_* & \Gamma_*^\top \\ \Gamma_* & 0 \end{bmatrix}, \quad \frac{\partial R_i}{\partial \bar{y}}(y^*; y^*) = \begin{bmatrix} E_* & 0 \\ 0 & 0 \end{bmatrix}$$

for $i = \mathrm{SCP}, \mathrm{SCQP}$. Analogous to the proof of Lemma 2 we know that the implicit function theorem is applicable and iteration operator $y_i^{\mathrm{sol}}(\bar{y})$ uniquely defined and continuously differentiable for $\bar{y}$ close to y^*. Furthermore its Jacobian is given by

$$\frac{\mathrm{d}y_i^{\mathrm{sol}}}{\mathrm{d}\bar{y}}(\bar{y}) = -\left(\frac{\partial R_i}{\partial y}(y_i^{\mathrm{sol}}(\bar{y}); \bar{y})\right)^{-1} \frac{\partial R_i}{\partial \bar{y}}(y_i^{\mathrm{sol}}(\bar{y}); \bar{y}) \tag{23}$$

for $i = \mathrm{SCP}, \mathrm{SCQP}$. At fixed point y^*, with $y_i^{\mathrm{sol}}(y^*) = y^*$, this evaluates to

$$\frac{\mathrm{d}y_i^{\mathrm{sol}}}{\mathrm{d}\bar{y}}(y^*) = \underbrace{-\begin{bmatrix} B_* & \Gamma_*^\top \\ \Gamma_* & 0 \end{bmatrix}^{-1} \begin{bmatrix} E_* & 0 \\ 0 & 0 \end{bmatrix}}_{:= A_*} \tag{24}$$

for $i = \mathrm{SCP}, \mathrm{SCQP}$. Now consider the iteration $y_{k+1} = y_i^{\mathrm{sol}}(y_k)$, with y_0 sufficiently close to y^*. Taylor expansion around y^* yields

$$y_{k+1} - y^* = A_*(y_k - y^*) + \mathcal{O}(\|y_k - y^*\|^2). \tag{25}$$

As is a standard result of linear stability analysis, convergence of y_k to y^* is then determined by the spectral radius $\rho(A_*)$. If $0 < \rho(A_*) < 1$ the sequence converges linearly with asymptotic contraction rate $\rho(A_*)$. When $\rho(A_*) = 0$ it converges faster than linear. If $\rho(A_*) = 1$ we cannot decide about convergence by considering only $\rho(A_*)$, and for $\rho(A_*) > 1$ it diverges locally.

Let us now derive a more convenient form of characterizing the spectral radius. As has been proven in [11, Thm. 4], the nonzero eigenvalues A_* coincide with the eigenvalues of $\widetilde{A}_* := \widetilde{B}_*^{-1}\widetilde{E}_*$. Due to $\widetilde{B}_* \succ 0$, the square root $\widetilde{B}_*^{-\frac{1}{2}}$ exists and is uniquely defined. Now

$$\widetilde{A}_{**} := \widetilde{B}_*^{\frac{1}{2}} \widetilde{A}_* \widetilde{B}_*^{-\frac{1}{2}} = \widetilde{B}_*^{-\frac{1}{2}} \widetilde{E}_* \widetilde{B}_*^{-\frac{1}{2}} \tag{26}$$

has the same eigenvalues as $\widetilde{A}_*$, since they are similar. From the rightmost expression in (26) we can see that $\widetilde{A}_{**}$ is symmetric, so its eigenvalues are real. Its spectral radius $\rho(\widetilde{A}_{**}) = \rho(\widetilde{A}_*) = \rho(A_*)$ is then given by the smallest α satisfying

$$-\alpha I \preceq \widetilde{B}_*^{-\frac{1}{2}} \widetilde{E}_* \widetilde{B}_*^{-\frac{1}{2}} \preceq \alpha I, \tag{27}$$

with I the identity matrix of appropriate size. From this, (21) directly follows and, recalling that $\widetilde{E}_* = \widetilde{\Lambda}_* - \widetilde{B}_*$, (22) is a direct consequence. Substituting $\alpha = 1$ into the left-hand side LMI of (22) we can see that $\widetilde{B}_* \succeq \frac{1}{2}\widetilde{\Lambda}_*$ is a necessary condition for convergence: if we had $\widetilde{B}_* \prec \frac{1}{2}\widetilde{\Lambda}_*$, the LMI could only hold for $\alpha > 1$, which means divergence. Finally, if $\widetilde{\Lambda}_* \succ 0$, we know that the right-hand side LMI of (22) has to hold for some $\alpha < 1$, and the same goes for the left-hand side LMI if $\widetilde{B}_* \succ \frac{1}{2}\widetilde{\Lambda}_*$ holds. Therefore $\widetilde{B}_* \succ \frac{1}{2}\widetilde{\Lambda}_*$ is sufficient for linear convergence. ∎

V. Further Considerations

We have now seen that both methods have the same local convergence rate. Also, both can be seen as generalizations of CGN. But of course the methods are not identical and there are more characteristics to consider. As a heuristic we propose that SCP shows better global convergence behaviour: both with respect to the number of iterations and in the sense of robustness, i.e., whether it converges at all. The reasoning is that in a certain sense SCP performs a better approximation of the original NLP, throwing away less information. SCQP keeps only up to second order information of objective and constraints, evaluated at the current iterate. Furthermore – though SCQP uses this curvature information in the Hessian approximation – it still linearizes the constraints. SCP on the other hand keeps full information about the convexities and thus has more accurate knowledge of the curvature when moving away from the current iterate.

The main advantage of SCQP is that in general it has cheaper iterations, since it only solves a QP. SCP in contrast has to solve a convex, though generally nonlinear, program at every iteration. This means that especially for initial guesses close to the solution SCQP is most likely faster. For general initial guesses it might still be faster, even though taking more iterations, but less likely to converge.

Another difference to consider is that SCP is a multiplier-free method, in the sense that the convex approximations only depend on the primal variable. SCQP on the other hand depends on the inequality multipliers at the solution of the current QP to construct the Hessian approximation for the next QP. It therefore needs a larger memory.

VI. Illustrative Example

We will now illustrate these results by applying them to a simple optimal control problem (OCP). Consider an unmanned aerial vehicle (UAV) with position $p = (p_x, p_z) \in \mathbb{R}^2$ and velocity $v = (v_x, v_z) \in \mathbb{R}^2$. We can control it via an omnidirectional thrust force $u = (u_x, u_z) \in \mathbb{R}^2$, which is constrained in magnitude, $u^\top u \leq \bar{u}$. Additionally the UAV is subject to drag and there is wind with velocity $v_\mathrm{w} \in \mathbb{R}^2$. Its nonlinear dynamics are then given by the first order ODE

$$\dot{x} = \psi(x, u) = \begin{bmatrix} v \\ g + u - d\|v - v_\mathrm{w}\|_2(v - v_\mathrm{w}) \end{bmatrix} \tag{28}$$

with state $x = (p, v)$, gravity vector $g = (0, -G)$, gravitational constant G, and drag constant d. We denote by $\Psi(x, u)$ one step of explicit fourth order Runge-Kutta integration over time step Δt.

The control goal is to stabilize the UAV at the reference position p_ref, corresponding to reference state $x_\mathrm{ref} = (p_\mathrm{ref}, 0)$, starting at initial state $\bar{x}_0 = (\bar{p}_0, \bar{v}_0)$. Furthermore there is a mountain, above which the UAV has to stay. The mountain is modeled as parabola, which leads to the constraint

$$h(p) := \bar{\zeta} - \beta(p_x - \bar{\chi})^2 - p_y \leq 0, \tag{29}$$

where $(\bar{\chi}, \bar{\zeta})$ is the peak of the mountain and $\beta < 0$ determines its steepness. The time horizon for this endeavor is T, which we discretize into N intervals, such that we have the discrete time step $\Delta t = T/N$. By using piecewise constant controls $u_k \in \mathbb{R}^2$, we can express this as the NLP

$$\begin{aligned}
\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad & \sum_{k=0}^{N-1} \|x_k - x_\mathrm{ref}\|_Q^2 + \|u_k\|_R^2 \\
& \qquad + \|x_N - x_\mathrm{ref}\|_{Q_N}^2 && \text{(30a)} \\
\text{s.t.} \quad & x_0 = \bar{x}_0, && \text{(30b)} \\
& x_{k+1} = \Psi(x_k, u_k), \quad k = 0, \dots, N-1, && \text{(30c)} \\
& u_k^\top u_k \leq \bar{u}^2, \quad k = 0, \dots, N-1, && \text{(30d)} \\
& h(p_k) \leq 0, \quad k = 0, \dots, N, && \text{(30e)}
\end{aligned}$$

with weighting matrices $Q, Q_N \in \mathbb{R}^{4\times4}$ and $R \in \mathbb{R}^{2\times2}$. As default parameters we use $\bar{x}_0 = 0$, $p_\mathrm{ref} = (10, 0)$, G = 9.81, $\bar{u} = 2G$, $d = 0.05$, $w = (-1, 0)$, $\bar{\chi} = 5$, $\bar{\zeta} = 2.5$, $\beta = 0.25$, $T = 2.5$ and $N = 50$. Furthermore $Q = \mathrm{diag}(100, 100, 0.1, 0.1)$, $R = \mathrm{diag}(0.1, 0.1)$ and $Q_N = \mathrm{diag}(100, 100, 100, 100)$. Fig. 1 visualizes an optimal trajectory for this setting.

Both methods are implemented in CasADi [1], a symbolic framework for numerical optimization, which computes all necessary derivatives via Algorithmic Differentiation and provides an interface to several optimization solvers. The convex problem arising from SCP is solved with IPOPT [12], an interior point method for general NLP, using MA57 as linear solver [6]. Note that this is a rather naive choice, as it is not exploiting the fact that in this example the CP is actually a convex Quadratically Constrained Quadratic Program (QCQP). The QP from SCQP is solved with HPMPC [5], a fast interior point QP solver tailored to problems arising from Model Predictive Control. As convergence criterion for both methods the primal-dual step $z_k - z_{k-1}$ is used. If $\|z_k - z_{k-1}\|_\infty \leq \epsilon_\mathrm{tol}$ for some tolerance $\epsilon_\mathrm{tol} > 0$, the algorithms are considered as converged.

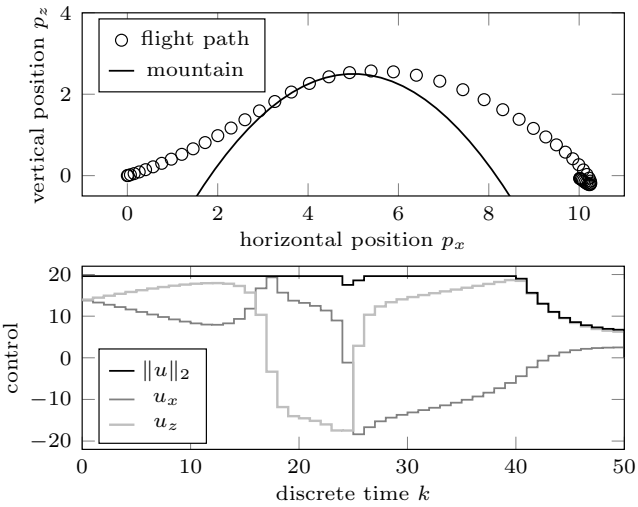

Fig. 1. Top: position trajectory of solution. Bottom: control trajectory of solution.

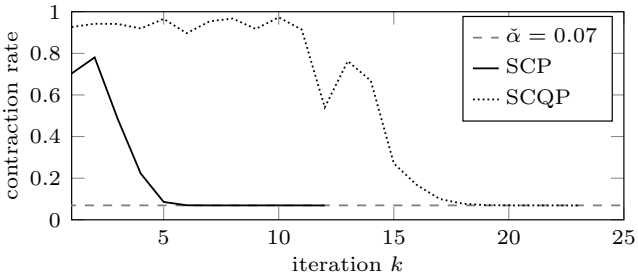

Fig. 2. Empirical contraction rate of SCP and SCQP when starting at initial guess $z_0 = 0$ compared to theoretical asymptotical rate $\breve{\alpha}$ computed from the LMI.

To investigate the local contraction rate, the problem is solved with default parameters and initial guess $z_0 = 0$. The resulting empirical contraction rate at iteration k is computed as

$$\kappa_k = \frac{\|z_k - z^*\|_2}{\|z_{k-1} - z^*\|_2}. \tag{31}$$

In Fig. 2 the empirical contraction rates of SCP and SCQP are compared to the theoretical local contraction rate $\breve{\alpha}$ obtained from LMI (21). It can be observed that for the last of handful of iterations, i.e., the region of local convergence, the observed rates match the theoretical result quite well.

To illustrate the global convergence behaviour we create a collection of different – though similar – problems by

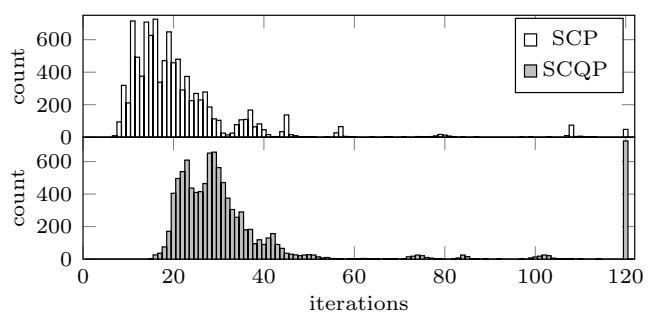

Fig. 3. Histogram of the number of iterations needed for solving 100 random variations of problem (30) up to tolerance $\epsilon_{\text{tol}} = 10^{-5}$, with 100 random initial guesses each. The rightmost bar contains all problems which took 120 iterations or more to solve.

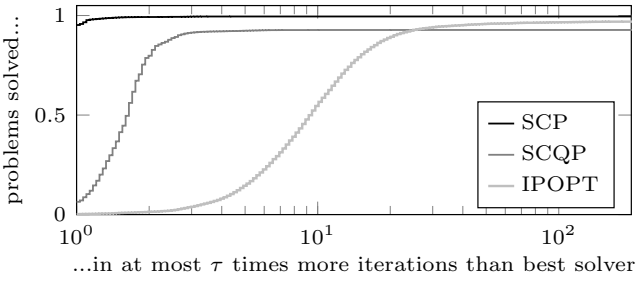

Fig. 4. Performance profile of the methods for solving 100 random variations of problem (30) up to tolerance $\epsilon_{\text{tol}} = 10^{-5}$, with 100 random initial guesses each. At the right border the fraction of problems solved by SCP is around 0.995. For SCQP it is approximately 0.927 and for IPOPT around 0.969.

randomizing some of the parameters of problem (30). More specifically, the randomized parameters and their (uniform) distributions are $\bar{p}_0 \in [-2, 2)^2$, $p_{\text{ref}} \in [8, 12) \times [-2, 2)$, $d \in [0, 0.2)$, $\bar{\chi} \in [2.5, 7.5)$, $\bar{\zeta} \in [0, 6)$, $\beta \in [0, 1)$. We then draw 100 instantiations of these parameters, resulting in 100 problems. If for a problem the initial or target position would lie inside the mountain, its parameters are redrawn. We solve each of the problems for 100 random initializations of w, distributed uniformly in $[-10, 10)^{n_w}$. The dual variables are always initialized at 0. Fig. 3 shows a histogram of the resulting number of iterations. We can see that SCP both solves more of the problems and generally uses less iterations.

As another reference, each problem is also solved by IPOPT directly, without knowledge of the convex-over-nonlinear substructure. Note that iterations of interior point methods such as IPOPT are conceptually different from those of sequential approximation methods. The latter have an additional layer of inner iterations for solving the subproblem, which are not counted here. Nonetheless the iterations of IPOPT can provide a meaningful point of reference if interpreted with care. A further caveat is that IPOPT uses a more sophisticated convergence criterion than our implementation of SCP and SCQP. Therefore the meaning of convergence is not directly comparable, but variations of the IPOPT tolerance parameter within a reasonable range (10^{-2} to 10^{-10}) have shown that its influence on the number of iterations for this problem is neglectable (about 3 iterations). For the experiments a tolerance of 10^{-5} has been used.

A performance profile [4] of the three methods can be found in Fig. 4. As SCP mostly outperforms SCQP and IPOPT in terms of iterations, its performance profile is almost flat. For around 4.7% of the problems, SCQP takes the fewest iterations. It then grows faster than IPOPT, but flatlines having solved the least fraction of the problems. The iterations of IPOPT show the slowest growth, but at the end it has solved almost as many problems as SCP. Note again that comparing only the number of iterations does not give the full picture, especially considering that SCP has the most expensive iterations of the tratiohree methods. Comparing performance with respect to computation time would therefore most likely yield a result less in favour of SCP. But as a fair comparison of timings is much harder to obtain, we refrain from providing any results on this.

VII. Conclusions

In this paper we have shown that under some mild conditions SCP has the same local linear convergence rate as SCQP and how it can be tightly characterized. We further discussed some advantages and disadvantages of both algoritms and finally illustrated their behaviour using a numerical example.

References

[1] Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018.

[2] H. G. Bock. Recent advances in parameter identification techniques for ODE. In *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, pages 95–121. Birkhäuser, 1983.

[3] Moritz Diehl and Florian Messerer. Local convergence of generalized Gauss-Newton and sequential convex programming. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2019.

[4] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91:201–213, 2002.

[5] G. Frison, H. B. Sorensen, B. Dammann, and J. B. Jørgensen. High-performance small-scale solvers for linear model predictive control. In *Proceedings of the European Control Conference (ECC)*, pages 128–133, June 2014.

[6] HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk.

[7] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[8] S.M. Robinson. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms. *Mathematical Programming*, 7:1–16, 1974.

[9] Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

[10] Q. Tran-Dinh and M. Diehl. Local convergence of sequential convex programming for nonconvex optimization. In M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels, editors, *Recent advances in optimization and its application in engineering*, pages 93–103. Springer-Verlag, 2010.

[11] Robin Verschueren, Niels van Duijkeren, Rien Quirynen, and Moritz Diehl. Exploiting convexity in direct optimal control: a sequential convex quadratic programming method. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2016.

[12] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.