A variant to Sequential Quadratic Programming for Nonlinear Model Predictive Control

Giampaolo Torrisi, Sergio Grammatico, Roy S. Smith, Manfred Morari

Abstract—Sequential Quadratic Programming (SQP) denotes an established class of methods for solving nonlinear optimization problems via an iterative sequence of Quadratic Programs (QPs). Several approaches in the literature have established local and global convergence of the method. This paper considers a variant to SQP that, instead of solving each QP, at each iteration follows one gradient step along a direction that is proven to be a descent direction for an augmented Lagrangian of the nonlinear problem, and in turn is used to generate the next QP. We prove global convergence to a critical point of the original nonlinear problem via a line search that requires the same assumptions as the SQP method.

The method is then applied to nonlinear Model Predictive Control (MPC). To simplify the problem formulation and achieve faster convergence, we propose a locally convergent reformulation. An important speed-up is observed in practice via a specific initialization. The computational efficiency of the proposed method is finally shown in a numerical example.

I. INTRODUCTION

Model Predictive Control (MPC) strategies rely on the control inputs being determined as solutions to optimization problems. When the system dynamics are nonlinear, the corresponding optimization problem becomes non-convex.

Among the numerous approaches introduced in the literature to deal with Nonlinear Problems (NLPs), we focus on the Sequential Quadratic Programming (SQP), which consists of addressing the nonlinear problem via the solution of a sequence of (easier) Quadratic Programs (QPs) [1], [2], [3]. Each optimizer updates the next QP and, under some assumptions, convergence is shown to critical points of the original nonlinear problem, i.e. points satisfying at least the first order KKT conditions.

The basic SQP method is locally convergent to a local minimum at a rate that depends on the quality of the approximation of the Lagrangian Hessian. In fact, using exact Lagrangian Hessian yields quadratic rate of convergence, and extensive literature has focused on deriving easy-to-compute approximations to such Hessian that guarantee superlinear convergence [3], [4].

Two methods are employed in SQP to guarantee global convergence to a critical point of the original problems. The trust-region method includes an additional constraint on the optimization variable [3], [5]. Line-search methods, instead, consider merit functions that are globally decreased. The update equation includes an additional damping parameter that is derived via line search [6], [7], [8]. Because of this parameter, the convergence is slower, but holds globally.

In this paper, we consider a variant to SQP for nonlinear optimization problems. Instead of solving exactly every generated QP, we consider only one gradient step, that then directly updates the next QP. We prove global convergence to a critical point via a line-search argument, under the same assumptions of the SQP [3]. As in SQP the convergence speed is reduced by the damping parameter. Local convergence properties are also presented, that yield linear rate when the initialization is sufficiently close to the solution. This is often the case in nonlinear MPC, since a good warm-start comes from the previous optimization problem. We also introduce an heuristics that modifies the algorithm to achieve an important speed-up in the implementation.

The paper is structured as follows: the proposed method for optimization problems is generically described in Section II, along with a brief review of the Sequential Quadratic Programming method. The mathematical proof of convergence of the proposed algorithm is contained in Section III. In Section IV the method is applied to a nonlinear MPC problem, whose numerical results are shown and compared to commercial solvers in Section V. Finally, Section VI concludes the paper.

II. A VARIANT TO SEQUENTIAL QUADRATIC PROGRAMMING FOR NONLINEAR OPTIMIZATION PROBLEMS

We consider the inequality constrained nonlinear optimization problem (NLP)

\[
\begin{align*}
\min_{z \in \mathbb{R}^n} & \quad J(z) \\
\text{s.t.} & \quad g(z) \leq 0,
\end{align*}
\]

where the functions \( J : \mathbb{R}^n \to \mathbb{R} \) and \( g : \mathbb{R}^n \to \mathbb{R}^m \) are two times differentiable functions, possibly non-convex. Let us define the Lagrangian of the NLP in (1) as

\[ L(z, \lambda) := J(z) + g(z)^T \lambda, \]

where \( \lambda \in \mathbb{R}^m_+ \). We call \( z^* \) a critical point of (1) if it satisfies the following first order conditions.

A1. There exist \( \lambda^* \in \mathbb{R}^p \) such that

\[
\nabla L(z^*, \lambda^*) = \nabla J(z^*) + \nabla g(z^*) \lambda^* = 0,
\]

\[
0 \leq \lambda^* \perp g(z^*) \leq 0,
\]
where the notation \( a \perp b \) indicates that \( a_j b_j = 0 \) for all components \( j \). Further, strict complementarity holds.

A2. The matrix of the active inequality constraint gradients at a KKT point, \( \nabla g_a(z^*)^\top \), has full column rank.

### A. Sequential Quadratic Programming

The Sequential Quadratic Programming (SQP) method generates a sequence \( \{z(i)\}_i \) to determine a critical point \( z^* \) (or, under some additional assumptions, a strong local minimum) via the solution of a sequence of Quadratic Programs (QPs). In particular, given the current iterate \( z(i) \), the method generates the QP

\[
d(i)^i := \arg \min_{d(z)} \frac{1}{2} d(z)^\top H(i) d(z) + \nabla J(z(i))^\top d(z)
\]

s.t. \( g(z(i)) + \nabla g(z(i))^\top d(z) \leq 0 \)

and then the sequence gets updated as

\[
z(i+1)^i := z(i)^i + (t(i)^i) d(i)^i,
\]

where \( t(i) \in [0, 1] \) is a step size, and \( H(i) \) is either the exact Hessian of the Lagrangian \( L \) of (1), or an appropriate approximation, satisfying the following assumptions.

B1. The matrices \( \{H(i)\}_i \) are uniformly positive definite on the null spaces of the matrices \( \nabla g_a(x(i)^i) \), i.e., \( \exists \beta_1 > 0 \) such that, for all \( i \),

\[
d(z)^\top H(i) d(z) \geq \beta_1 \|d(z)\|^2 \quad \text{for all } d(z) \text{ s.t. } \nabla g_a(x(i)^i) d(z) = 0.
\]

B2. The matrices \( \{H(i)\}_i \) are uniformly bounded, i.e., \( \exists \beta_2 > 0 \) such that, for each \( i \), \( \|H(i)\| \leq \beta_2 \).

Several possible choices for the Hessian \( H(i) \) have been considered in the literature. By setting \( H(i) \) as the Hessian of the Lagrangian of (1) and unit step size, local convergence to the desired \( z^* \) is achieved with a quadratic rate. Other choices make the computation of \( H(i) \) easier, but deteriorate the convergence speed.

The use of inequality constraints requires the definition of a slack variable \( s(i) \geq 0 \) such that \( c(z(i), s(i)) := g(z(i)) + s(i) = 0 \). Similarly to (4), the variable \( s(i) \) is updated via the step size \( t(i)^i \) and search direction \( d(s)^i \), as follows:

\[
s(i+1)^i = s(i)^i + t(i)^i d(s)^i,
\]

with

\[
d(s)^i := - \left( g(z(i))^\top + \nabla g(z(i))^\top d(s)^i + s(i) \right)
\]

\[
= - \left( c(z(i), s(i)) + \nabla g(z(i))^\top d(s)^i \right).
\]

Note that, if \( s(i)^i \geq 0 \), an update \( t(i)^i \in [0, 1] \) guarantees \( s(i+1)^i \geq 0 \).

To ensure global convergence to a critical point, step sizes \( t(i)^i \) differ from 1 are employed in SQP, together with an augmented Lagrangian function, i.e.,

\[
\mathcal{L}_{\text{aug}}(z, s, \rho) := J(z) + c(z, s)^\top \lambda(z, s) + \frac{\rho}{2} \|c(z, s)\|^2_2,
\]

where \( \rho > 0 \) is a parameter to be determined and

\[
\lambda(z, s) := - (\nabla g(z)^\top \nabla g(z) + \text{diag}(s))^{-1} \nabla g(z)^\top \nabla J(z)
\]

are the least squares estimates of the optimal multipliers in (2) [7].

Then, the Wolfe conditions are typically used to determine the step length \( t(i)^i \) such that

\[
\mathcal{L}_{\text{aug}} \left( z(i)^i + t(i)^i d(s)^i, s(i)^i + t(i)^i d(s)^i, \rho \right) \leq \mathcal{L}_{\text{aug}} \left( z(i)^i, s(i)^i, \rho \right) + \sigma_1 t(i)^i \nabla \mathcal{L}_{\text{aug}} \left( z(i)^i, s(i)^i, \rho \right)^\top d(s)^i
\]

\[
\nabla \mathcal{L}_{\text{aug}} \left( z(i)^i + t(i)^i d(s)^i, s(i)^i + t(i)^i d(s)^i, \rho \right)^\top d(s)^i \geq \sigma_2 \nabla \mathcal{L}_{\text{aug}} \left( z(i)^i, s(i)^i, \rho \right)^\top d(s)^i,
\]

for some \( 0 < \sigma_1 < \sigma_2 < 1 \).

Under the following assumptions, for \( \rho \) large enough, \((d(s)^i, d(z)^i)\) from (1) can be proven to always be a descent direction for \( \mathcal{L}_{\text{aug}}(z(i)^i, s(i)^i, \rho) \), which implies that there always exists a step size \( t(i)^i \) that satisfies (9) [3]:

C1. There exists a compact set \( S \) that contains all the starting points and subsequent iterates.

C2. For all \( z \in S \), the QP in (3) is feasible and the active inequality constraint gradients are full-rank.

The SQP steps for the NLP in (1) are summarized in Algorithm 1.

### B. Proposed variant to Sequential Quadratic Programming

The SQP method presented in Section II requires the computation of the optimizer to each of the QPs in (3), hence each QP has to solved completely to determine the update \( d(z)^i \).

In this paper, we propose an alternative method to compute a critical point \( z^* \) for the NLP in (1), that does not require the exact solution of each generated QP.

Specifically, we propose to determine \( d(z)^i \) by one projected gradient step of the QP in (3) from an initialization \( v(i) \), as formalized next.

\[
d(z)^i := \Pi_{C(i)} \left( v(i) - s(i)^i \left( H(i) v(i) + \nabla J(z(i))^\top \right) \right),
\]

with gradient step size \( s(i)^i \) and \( \Pi_{C(i)}(\cdot) \) being the Euclidean projection onto the set \( C(i) := \{ d(z) \in \mathbb{R}^n \mid g(z(i))^\top + \nabla g(z(i))^\top d(z) = 0 \} \). As in the standard gradient method, easy-to-project constraints need to be considered to make the projection inexpensive.

For the initialization \( v(i) \) we consider two options, either \( v(i) = 0 \), or the previously determined \( d(z)^{(i-1)} \), since we expect the new update \( d(z)^i \) to be relatively close to \( d(z)^{(i-1)} \), based on the following test:

\[
v(i) = \begin{cases} 
  d(z)^{(i-1)} & \text{if } p(i)(d(z)^{(i-1)}) < 0 \\
  0 & \text{otherwise}
\end{cases}
\]

where

\[
p(i)(d(z)^{(i-1)}) := (d(z)^{(i-1)})^\top \left( I - \alpha(i) H(i) \right) d(z)^{(i-1)}
\]

\[
d(z)^{(i-1)} := \Pi_{C(i)} \left( d(z)^{(i-1)} - \alpha(i) H(i) d(z)^{(i-1)} + \nabla J(z(i))^\top \right).
\]
Algorithm 1 Sequential Quadratic Programming.

procedure SQP FOR NLP (1)
Initialize $i = 0$ and $z(0)$
repeat
Compute $d_z^{(i)}$ as the solution to (3)
Determine the step size $t^{(i)}$ that satisfies (9)
e.g. via line search
Compute $d_s^{(i)}$ from (6)
Update $z^{(i+1)}$ and $s^{(i+1)}$ as in (4) and (5)
$i ← i + 1$
until Convergence
return $z(i)$
end procedure

This allows one to obtain a steeper descent whenever $p^{(i)}(d_z^{(i-1)}) < 0$ (see Theorem 3.2). From (11), (12) it follows that
$$p^{(i)}(d^{(i)}) \leq 0. \tag{13}$$
Once the value $d_z^{(i)}$ is computed from (10), the variable $z^{(i+1)}$ gets updated according to (4) where, as in the SQP method, the step size $t^{(i)}$ is found via line search to satisfy (9). The proposed approach is summarized in Algorithm 2.

III. PROOF OF CONVERGENCE OF THE PROPOSED ALGORITHM

In this section, we show the convergence properties of the proposed approach in Algorithm 2 under the same assumptions of the SQP method. The analysis follows the classical SQP proof, that is, we show that the algorithm converges to a critical point through the decrease of the augmented Lagrangian merit function $L_{aug}$ in (7). First we need to slightly adapt Assumption C2 to the update equation considered, in particular:

C3. For all $z \in S$, the gradient step in (10) is feasible and the active inequality constraint gradients are full-rank. Clearly, feasibility of the QP in (3) implies feasibility of (10), as the feasible set of each coincides.

We start with the characterization of the critical points and local minima of $L_{aug}(\cdot, \rho)$ via the following theorem.

Theorem 3.1 ([7, Propositions 2 and 3]): Assume B1, B2, C1 and C3 are satisfied. Then for $\rho$ sufficiently large, the following hold:

(i) $(z^*, s^*)$, with $s^* = -g(z^*) \geq 0$, is a critical point of $L_{aug}(\cdot, \rho)$ if and only if $z^*$ is a critical point of (1);

(ii) $z^*$ is a strong local minimum of $L_{aug}(\cdot, \rho)$, with $s^* = -g(z^*) \geq 0$, if and only if $z^*$ is a strong local minimum of (1); \hfill $\square$

We then proceed with our main theoretical results, namely that the update $(d_z, d_s)$ defined according to (10) and (6) is a descent direction for $L_{aug}(z, s, \rho)$.

Proof: $(d_z, d_s)$ is a descent direction for $L_{aug}(z, s, \rho)$ if and only if $d_z^T \nabla L_{aug}(z, s, \rho) < 0$. Note that the gradient $\nabla L_{aug}(z, s, \rho)$ consist of two parts, i.e.: $d_z^T \nabla L_{aug}(z, s, \rho) = \nabla_x L_{aug}(z, s, \rho) c(z, s)$

We first consider the term $d_z^T \nabla x L_{aug}(z, s, \rho)$. Note that the projection in (10) can be recast as a quadratic program, whose KKT conditions by C3 yield:

$$\frac{1}{\alpha} (d_z - (I - \alpha H) v - \alpha \nabla J(z)) + \nabla g(z) \mu = 0 \tag{15}$$

with the last inequality following from (6) and (13). By the complementarity condition in (15), again combined with (6), it follows that $(s + d_s)^T \mu = 0$, and since $\mu \geq 0$ and $s \geq 0$, then $d_s^T \mu \leq 0$. Thus, by defining $a(z, s) :=$
\[ \mu + \nabla_z \lambda(z, s)^T d_z + \nabla_s \lambda(z, s)^T d_s - \lambda(z, s) \text{ in (14):} \]
\[ d_z^T \nabla_z L_{aug}(z, s, \rho) + d_s^T \nabla_s L_{aug}(z, s, \rho) \]
\[ = -\frac{1}{\alpha} d_z^T d_z + c(z, s)^T a(z, s) - \rho c(z, s)^T c(z, s) \]
\[ \leq -\frac{1}{\alpha} d_z^T d_z + \|c(z, s)\|_2 \|a(z, s)\|_2 - \rho \|c(z, s)\|_2^2. \]

If \( c(z, s) \neq 0, \) by choosing \( \rho \geq \|a(z, s)\|_2 / \|c(z, s)\|_2, \) the second and third terms of the expression above are made non-positive. If \( d_z = 0 \) and \( c(z, s) = 0, \) the test in (11) yields \( v = 0. \) Therefore, the conditions in (15) become analogous to (2), with \( z = z \) and \( \lambda^* = \mu. \) By (6), \( s = -g(z) \) yields that, by C3, \( (z, s) \) is a critical point for (1).

We exploit the previous theorem to establish convergence to a critical point of \( L_{aug}(\cdot, \rho). \)

**Theorem 3.3:** If \( \rho > 0 \) is chosen such that
\[ -\frac{\partial^2}{\partial z \partial s} L_{aug}(z(s), s(s), \rho) \]
\[ > 0, \quad (16) \]
then the iteration in Algorithm 2 with step length chosen to satisfy the Wolfe conditions in (9) is globally convergent to a critical point of \( L_{aug}(\cdot, \rho). \) \( \Box \)

The condition in Theorem 3.3 is guaranteed by choosing a sufficiently large \( \rho. \) The proof follows [3, Proof of Theorem 10] and consists of showing that
\[ \lim_{i \to \infty} \nabla L_{aug}(z(s), s(s), \rho) = 0, \]
hence convergence to a critical point. As computing \( \|a(z, s)\|_2 \) analytically is impractical, the value of \( \rho \) should be determined numerically.

Note that, as in the standard SOP with line search, a critical point for \( L_{aug}(\cdot, \rho) \) need not be a local minimum of (1), see [3] for a discussion on the convergence properties.

**IV. APPLICATION TO NONLINEAR MPC**

A parameter \( \rho \) that satisfies (16) at every iteration of the SOP algorithm is typically not known a priori. The option of tuning \( \rho \) during the iterations of the algorithm has long been discussed, e.g. by increasing \( \rho \) when (16) is not satisfied [9], [7], [8]. A large value for \( \rho \) can cause computational difficulties, as it makes the term \( J(z) \) in (7) negligible. Other approaches forgo the satisfaction of (16) when the test fails, and design \( t \) to reduce the constraint violation [6].

In this section we present two locally convergent versions of the proposed algorithm, specifically designed for nonlinear MPC problems, that exploit the quadratic cost function and the structure of the constraints to avoid the need for an augmented Lagrangian function, and as a result, take a full step size \( t^{(i)} = 1 \) \( \forall i. \) The distinction between the two versions is in the initialization \( v^{(i)}: \) taking \( v^{(i)} = 0 \) gives theoretical guarantees of local convergence (see Theorem 4.1). On the other hand, an initialization \( v^{(i)} = 0 \) based on a specific test on a previous iterate \( d_z^{(i-1)} \) gives better practical performance, albeit without theoretical guarantees (see next section).

For the MPC formulation we consider the following general nonlinear discrete-time optimal control problem without state constraints:
\[ \min_{x_{k+1}, u_k} \sum_{k=0}^{N-1} \left\{ \frac{1}{2} x_k^\top Q x_k + \frac{1}{2} u_k^\top R u_k \right\} + \frac{1}{2} x_N^\top P x_N \]
\[ \text{s.t.} \quad x_{k+1} = f(x_k, u_k) \quad \forall k \in \mathbb{Z}[0, N - 1] \]
\[ u_k \in [a_k, b_k] \quad \forall k \in \mathbb{Z}[0, N - 1], \]
(17)

in which \( a_k < b_k \in \mathbb{R}^n \) and \( Q, R, P \geq 0. \)

For ease of notation, let us first define the vectors
\( u := [u_0; \ldots; u_{N-1}] \)
for the control input sequence, with bounds
\( a := [a_0; \ldots; a_{N-1}] \) and \( b := [b_0; \ldots; b_{N-1}], \)
and the corresponding state evolution \( x := [x_1; \ldots; x_N], \)
and stack the state and input cost matrices
\( Q = \text{blockdiag}(Q, \ldots, Q, P) \)
and \( R = \text{blockdiag}(R, \ldots, R). \)
Then the MPC problem in (17) in compact form is
\[ \min_{x, u} \frac{1}{2} x^\top Q x + \frac{1}{2} u^\top R u =: J(x, u) \]
\[ \text{s.t.} \quad x = \psi(u), \quad u \in [a, b], \]
(18)

where for a fixed initial state \( x_0 \) and given a sequence of inputs \( u \) as argument, \( \psi(u) \) is the predicted state sequence \( x \) according to the nonlinear dynamics \( x_{k+1} = f(x_k, u_k). \)

Starting from an initial guess \( u^{(i)} := [u_0^{(i)}; \ldots; u_{N-1}^{(i)}] \)
for the control input sequence at time step \( k \) and corresponding state evolution \( x^{(i)} := [x_1^{(i)}; \ldots; x_N^{(i)}], \)
we consider the vectors \( d_x^{(i)} \) and input \( d_u^{(i)} \), obtained via linearization of the dynamics \( x = \psi(u) \):
\[ x^{(i)} + d_x^{(i)} = \nabla \psi(u^{(i)})^\top d_u^{(i)} + \psi(u^{(i)}). \]
(19)

The vectors \( d_x^{(i)} \) and input \( d_u^{(i)} \) represent the perturbation from the nominal trajectory \( (x^{(i)}, u^{(i)}) \), and the \((h, j)\) block element of the matrix \( \nabla \psi(u^{(i)}) \) is
\[ [\nabla \psi(u)]_{h,j} = \left( \prod_{s=j}^{h-1} F_s^{(i)} \right)^\top G_j^{(i)}, \]
with \( F_j := \frac{df}{dx}(\psi_j(u), u_j), \)
\( G_j := \frac{df}{du}(\psi_j(u), u_j) \)
and the operator product \( \prod \) being 1 if \( j > h - 1. \)

Now, given \( (x^{(i)}, u^{(i)}) \), we consider the QP:
\[ \min_{d_x, d_u} \frac{1}{2} d_x^\top Q d_x + x^{(i)}^\top Q d_x \]
\[ + \frac{1}{2} d_u^\top R d_u + u^{(i)}^\top R d_u \]
\[ \text{s.t.} \quad d_x = \nabla \psi(u^{(i)})^\top d_u + \psi(u^{(i)}) - x^{(i)}, \]
\[ u^{(i)} + d_u \in [a, b]. \]
(20)

A computationally inexpensive choice for the Hessian matrices satisfying B1, B2 is obtained via Gauss-Newton approximation, and results in \( Q^{(i)} = Q \) and \( R^{(i)} = R \) [5], [10]. Further, let us include the equality constraint directly into the
cost function and obtain
\[
\min_{d_u} \frac{1}{2} d_u^\top H(u^{(i)}) d_u + \zeta(u^{(i)})^\top d_u
\]
\[\text{s.t. } u^{(i)} + d_u \in [a, b],\]
where
\[
H(u^{(i)}) := R + \nabla\psi(u^{(i)}) Q \nabla\psi(u^{(i)})^\top \geq 0,
\]
\[
\zeta(u^{(i)}) := R u^{(i)} + \nabla\psi(u^{(i)}) Q \nabla\psi(u^{(i)}),
\]
and the cost function is defined as
\[
q(d_u, u^{(i)}) := \frac{1}{2} d_u^\top H(u^{(i)}) d_u + \zeta(u^{(i)})^\top d_u.
\]

1) Sequential Quadratic Programming: The locally convergent SQP approach takes \(d_u^{(i)}\) as the solution to (21) and updates the state-input trajectory as follows:
\[
\begin{align*}
\mathbf{u}^{(i+1)} &= u^{(i)} + d_u^{(i)} \\
\mathbf{x}^{(i+1)} &= \nabla\psi(u^{(i)})^\top d_u^{(i)} + \psi(u^{(i)}).
\end{align*}
\]

2) Proposed Algorithm: Instead of solving (21) to optimality, we take only one gradient step from the initialization \(\mathbf{v}^{(i)}\), that is:
\[
d_u^{(i)} := \Pi_{[a,b]-u^{(i)}} \left( \mathbf{v}^{(i)} - s^{(i)} \left( H(u^{(i)})\mathbf{v}^{(i)} + \zeta(u^{(i)}) \right) \right)
\]
and update the state and input trajectory according to (23). Note that the input box constraints result in an inexpensive clipping.

As in the SQP, for the local convergence results of the algorithm we assume that the initialization \(z^{(0)} = (x^{(0)}, u^{(0)})\) is sufficiently close to a local minimum \(z^* = (x^*, u^*)\) and it is such that the correct active set at the solution is known [3]. The next theorem establishes the convergence of the sequence \(z^{(i)} = (x^{(i)}, u^{(i)})\) for the case of null initialization \(\mathbf{v}^{(i)} = 0\).

**Theorem 4.1** ([11]): Let A1, A2, C1 and C3 hold and assume that \((z^*, \lambda^*)\) is a critical point of (18) such that \(\nabla^2 \mathcal{L}(z^*, \lambda^*)\) is positive definite, and let the initialization \(z^{(0)}\) be close enough to \(z^*\). Then, there exist positive step sizes \((s^{(i)})_i\) such that the sequence \((z^{(i)})_i\) generated by (24) and (23) with \(\mathbf{v}^{(i)} = 0\) converges to \(z^*\) with linear rate. \(\square\)

The second formulation takes a different initialization \(\mathbf{v}^{(i)}\) depending on a test on the previous iterate \(d_u^{(i-1)}\).
\[
\mathbf{v}^{(i)} := \begin{cases} 
\Pi_{[a,b]-u^{(i)}} \left( d_u^{(i-1)} \right) & \text{if } q \left( \Pi_{[a,b]-u^{(i)}} \left( d_u^{(i-1)} \right), \mathbf{x}^{(i)}, u^{(i)} \right) < 0, \\
0 & \text{otherwise}.
\end{cases}
\]

This approach is motivated by the following heuristic: the SQP in (21) minimizes the function \(q(d_u, u^{(i)})\) with respect to \(d_u\), subject to the input constraint. Choosing an initialization \(\mathbf{v}^{(i)} = 0\) clearly yields \(q(\mathbf{v}^{(i)}, u^{(i)}) = 0\). If the previously computed \(d_u^{(i-1)}\) is such that \(q(\mathbf{v}^{(i)}, u^{(i)}) < 0\), then this initialization heuristic aims to accelerate the convergence.

### A. Linear algebra formulation

The algorithm step in (24) can be computed efficiently by exploiting sparsity of the matrices involved. First, we note that the Hessian \(H(u^{(i)})\) is never explicitly required, only the product \(H(u^{(i)}) \mathbf{v}^{(i)}\). Therefore, by the definition in (22), this product can be computed as follows. The matrix \(\nabla\psi(u^{(i)})^\top\) is block lower triangular matrix and every row group can be calculated iteratively from the one above, hence
\[
\mathbf{v} = \begin{bmatrix} v_0 \vline v_1 \vline \cdots \vline v_l \end{bmatrix} := \nabla\psi(u^{(i)})^\top \mathbf{v} = \begin{bmatrix} G_0 v_0 + \cdots + G_l v_l \end{bmatrix}.
\]
The product \(H(u^{(i)}) \mathbf{v}^{(i)}\) is:
\[
H(u^{(i)}) \mathbf{v}^{(i)} = R \mathbf{v}^{(i)} + \nabla\psi(u^{(i)}) Q \mathbf{v}^{(i)}
\]
where, again, the structure of \(\nabla\psi(u^{(i)})\) and of the block diagonal matrices \(Q\) and \(R\) is exploited.

For the calculation of the step size \(s\) a standard Armijo rule requires only evaluations of the objective function \(q\) and its derivative [12].

### V. Example: Model Predictive Control of an Inverted Pendulum

As a case study, we consider an inverted pendulum of length \(l\) with mass \(m\) concentrated at the tip and no friction acting on the cart and swing. The mass of the cart is \(M\) and the gravitational acceleration \(g\). The states \(x_1\) and \(x_2\) are respectively the cart position and velocity and \(x_3\) and \(x_4\) the pendulum angle and angular velocity. The input \(u\) is the applied force on the cart and it is subject to box constraints. We discretize the continuous-time dynamics,
\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{mg \sin x_3 \cos x_3 - ml x_2^2 \sin x_3 + u}{M + m \sin^2(x_3)} \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= g \frac{\sin x_3 \cos x_3 + u \cos x_3 - ml x_2^2 \sin x_3 \cos x_3}{l (M + m \sin^2 x_3)},
\end{align*}
\]
in the MPC model with explicit Euler method with sampling time \(T_s = 0.077\) s, and hence obtain an MPC problem of the form (17), with model Jacobians computed symbolically and prediction horizon \(N = 8\). The plant model is simulated in MATLAB with ode45 integration.

The system has two sets of unforced equilibria, the unstable ones \([p; 0; 2k\pi; 0]\), and the stable ones \(x = [p; 0; \pi + 2k\pi; 0]\), with \(p \in \mathbb{R}\) and \(k \in \mathbb{Z}\). Physically, the former correspond to the pendulum in the upright position, while the latter in the stable upside-down configuration. The goal of the controller is to stabilize the system around the origin, that is, to the unstable equilibrium, starting from the stable one \(x_0 := [0; 0; \pi; 0]\).
In Fig. 1 it shown the difference in the closed-loop cost obtained when solving the MPC problem in (17) to the full nonlinear solution or with a one-step SQP, that is, only one QP in (21) is solved at every time step, inspired to the Real-Time Iteration [10]. Since the closed-loop dynamics are significantly different, we consider other full NLP solutions as benchmarks to the proposed methods. In particular, along with the two versions of Section IV based on the initialization $\mathbf{v}^{(i)}$, we consider the commercial NLP solvers: SNOPT (Sequential Quadratic Programming) and the MPC-tailored FORCES PRO (Interior Point Method) [13], [14]. As the solutions and the closed-loop dynamics coincide for the initialization, the main difference is in the computational times.

A. Computational times

The average and worst case computational times obtained in the simulations of Figure 1 are given in Table I. The two versions of the proposed algorithm have been coded in C (sequential code) and compiled using Visual Studio 2015 compilers, enabling the \texttt{/Ox} optimization feature. The calculations have been performed on a Windows PC with processor Intel Core i7-3740QM 2.70Ghz. Both the variants to the proposed methods are in general superior to SNOPT for the specific problem. The variant with non-zero initialization yields an important speed-up and outperforms FORCES PRO.

<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. time (ms)</th>
<th>Best (ms)</th>
<th>Worst (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Alg. (v = 0)</td>
<td>0.77</td>
<td>6.5e−3</td>
<td>3.09</td>
</tr>
<tr>
<td>Proposed Alg. (v ≠ 0)</td>
<td>0.063</td>
<td>5.3e−3</td>
<td>0.24</td>
</tr>
<tr>
<td>SQP (SNOPT)</td>
<td>96.6</td>
<td>31.2</td>
<td>249</td>
</tr>
<tr>
<td>IP (FORCES PRO)</td>
<td>0.13</td>
<td>0.10</td>
<td>0.25</td>
</tr>
</tbody>
</table>

VI. OUTLOOK AND CONCLUSION

The paper introduces a variant to SQP to address nonlinear optimization problems. The main difference consists of not solving each QP, but just performing one step in the descent direction. The resulting vector updates the sequence $(z^{(i)})$, that is proven to converge to a critical point of the original nonlinear problem via typical line-search SQP arguments, including the same technical assumptions.

Since for real-time applications local faster convergence is often preferred, we presented a locally convergent formulation that does not require merit functions. The algorithm is tested for the MPC control of an inverted pendulum, along with commercial solvers. Sparsity eases the linear algebra involved to achieve competitive computational times.

REFERENCES