2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

An Almost Feasible Sequential Linear Programming Algorithm

David Kiessling[1], Charlie Vanaret[3], Alejandro Astudillo[1], Wilm Decré[1], Jan Swevers[1]

Abstract— **This paper proposes an almost feasible Sequential Linear Programming (afSLP) algorithm. In the first part, the practical limitations of previously proposed Feasible Sequential Linear Programming (FSLP) methods are discussed along with illustrative examples. Then, we present a generalization of FSLP based on a tolerance-tube method that addresses the shortcomings of FSLP. The proposed algorithm afSLP consists of two phases. Phase I starts from random infeasible points and iterates towards a relaxation of the feasible set. Once the tolerance-tube around the feasible set is reached, phase II is started and all future iterates are kept within the tolerance-tube. The novel method includes enhancements to the originally proposed tolerance-tube method that are necessary for global convergence. afSLP is shown to outperform FSLP and the state-of-the-art solver IPOPT on a SCARA robot optimization problem.**

I. Introduction

In nonlinear model predictive control, the solution of a nonlinear optimization problem must be computed in every sampling period so that the system is given an optimized control signal. Issues arise when no solution is available at the end of the sampling period. One option to overcome this difficulty is feasible algorithms that guarantee the feasibility of every iterate during the solution process and allow the algorithm to return a suboptimal, but feasible solution.

A successful feasible algorithm is FSQP [14], a line-search Sequential Quadratic Programming (SQP) algorithm that solves three subproblems in every iteration to keep the iterates feasible. First, a standard Quadratic Problem (QP) and a modified QP are solved to obtain a feasible descent direction that is close to the standard QP direction, then a second-order correction is performed. An arc search along the combination of both directions yields a feasible direction with a sufficient decrease in the objective function. Notably, the algorithm cannot treat nonlinear equality constraints directly and has to rely on ℓ_1 relaxed constraints instead [11]. Additionally, if no feasible initial guess is provided, the algorithm starts with a feasibility phase that attempts to find a feasible point.

A recent development is the SEQUOIA solver [8] that reformulates the general nonlinear optimization problem as a bilevel optimization problem, based on a residual optimization problem. This algorithm also first solves a feasibility problem, which determines an upper bound on the optimal solution. Then, an infeasible, non-tight lower bound of the optimal solution is generated. A bisection search is then performed until a given optimality gap is fulfilled.

In [13] the algorithmic framework FP-SQP was introduced: it includes a trust-region QP step that is projected onto the feasible set. No particular projection strategy is described, however certain properties should hold at the feasible iterate for the algorithm to globally converge. FP-SQP was successfully used for nonlinear Optimal Control Problems (OCP) with linear constraints in [10], in which a stabilizing feasibility procedure is based on the linear-quadratic regulator gain. The proposed projection works well for the considered OCPs but does not generalize to nonlinear constraints.

Another application of FP-SQP was proposed in [7]: a Feasible Sequential Linear Programming (FSLP) algorithm solves trust-region Linear Problems (LPs) and achieves feasible iterates with inner feasibility iterations that are based on iterated second-order corrections. An Anderson acceleration scheme improved the feasibility iteration algorithm in [6]. Several practical deficiencies of the FSLP algorithm were determined during extensive testing. In particular, the requirement to start the algorithm from a feasible point caused difficulties for complex OCPs. Moreover, the LPs in the feasibility iterations could be infeasible, which forces the algorithm to decrease the trust-region radius. Overall, the feasibility iterations mainly converge for small trust-region radii, which produces small steps for the FSLP algorithm.

A. Contributions

The contributions of this paper are threefold. Firstly, the practical deficiencies of the FSLP algorithm are demonstrated using illustrative examples. Secondly, we introduce a relaxation of the feasible algorithm based on the tolerance-tube method [15], called afSLP. The iterates are not forced to be feasible, but should stay in a tolerance-tube around the feasible set. This allows for early termination of the feasibility iterations, which reduces the overall computational load. Additionally, as in [8], [14], a feasibility phase allows for initialization at infeasible points. The phase terminates once an iterate inside of the tolerance-tube is found. Thirdly, we extend the tolerance-tube method with two novel features in order to guarantee global convergence. Finally, the improved performance of the almost feasible algorithm is demonstrated in practical examples. The tolerance-tube algorithm was implemented in C++ within CasADi [1].

B. Outline

This paper is structured as follows. In Section II, the original feasible sequential linear programming algorithm is

[1]MECO Research Team, Dept. of Mechanical Engineering, KU Leuven and Flanders Make@KU Leuven, 3001 Leuven, Belgium. {david.kiessling, alejandro.astudillovigoya, jan.swevers}@kuleuven.be

[3]Mathematical Algorithmic Intelligence Division, Zuse-Institut Berlin, Berlin, Germany. vanaret@zib.de

This work has been carried out within the framework of the Flanders Make SBO project DIRAC: Deterministic and Inexpensive Realizations of Advanced Control.

Copyright ©2024 EUCA

reviewed. Section III discusses the practical limitations of FSLP. A relaxed feasibility algorithm based on a tolerance-tube is presented in Section IV and simulation results are presented in Section V. Section VI concludes this paper.

C. Notation

In order to simplify the presentation, we change the problem formulation compared to [6], [7]. The general Nonlinear Problem (NLP) is given by:

$$\min_{w \in \mathbb{R}^{n_w}} \; f(w) \quad \text{s.t. } g(w) = 0, \; h(w) \leq 0, \tag{1}$$

where $w \in \mathbb{R}^{n_w}$, and $f\colon \mathbb{R}^{n_w} \to \mathbb{R}$, $g\colon \mathbb{R}^{n_w} \to \mathbb{R}^{n_g}$ and $h\colon \mathbb{R}^{n_w} \to \mathbb{R}^{n_h}$ are twice continuously differentiable. The gradient of f at w is given by $\nabla f(w)$. The Jacobian matrices of g and h are represented by $J_g(w) \in \mathbb{R}^{n_g \times n_w}$ and $J_h(w) \in \mathbb{R}^{n_h \times n_w}$, respectively. The iteration indices are denoted by superscripts, e.g., $w^{(1)} \in \mathbb{R}^{n_w}$ whereas vector components are specified by subscripts, e.g., $w_1 \in \mathbb{R}$. For a given iterate $w^{(k)}$, quantities evaluated at that iterate will be denoted with the same superscript, e.g., $f^{(k)}$ or $\nabla f^{(k)}$.

The measure of infeasibility is defined by $v(w) := \|g(w)\|_\infty + \|[h(w)]^+\|_\infty$, where $[h(w)]^+ := [\max\{h(w)_i, 0\}]_{i=1}^{n_h}$. The feasible set is denoted by $\mathcal{F} := \{w \in \mathbb{R}^{n_w} \mid g(w) = 0, h(w) \leq 0\}$.

II. Feasible Sequential Linear Programming

In this section, FSLP (Algorithm 1) is briefly introduced. The feasibility iterations are described in Algorithm 4. For a detailed discussion, see [7] and for an extension [6].

A. Outer Algorithm

At every iteration, (1) is linearized at the point $w^{(k)} \in \mathcal{F}$. The resulting trust-region LP is:

$$\begin{aligned} \min_{w \in \mathbb{R}^{n_w}} \quad & m_f^{(k)}(w) := (\nabla f^{(k)})^\top (w - w^{(k)}) \\ \text{s.t.} \quad & g^{(k)} + J_g^{(k)}(w - w^{(k)}) = 0, \\ & h^{(k)} + J_h^{(k)}(w - w^{(k)}) \leq 0, \\ & ||P(w - w^{(k)})||_\infty \leq \Delta^{(k)}. \end{aligned} \tag{2}$$

Here, $P \in \mathbb{R}^{n_y \times n_w}$ denotes a projection and (optionally) scaling matrix that selects the variables involved in the trust region. The identity matrix is usually used. The solution of (2), $\bar{w}^{(k)}$, is projected onto the feasible set using the feasibility iterations described in Algorithm 4. The feasible trial iterate is denoted by $\hat{w}^{(k)}$. If no feasible point can be found, the trust-region radius is decreased in (2). The termination criterion is $m_f^{(k)}(\bar{w}) = 0$: if no decrease in the model of the objective function m_f at a feasible point is possible, an optimal point was found and the algorithm terminates.

The standard trust-region update strategy and the acceptance test are given in Algorithm 2. As usual in trust-region algorithms, the quotient of actual and predicted reduction

$$\rho_{\text{II}}^{(k)} = \Delta f^{(k)} / \Delta m_f^{(k)} \tag{3}$$

Algorithm 1: FSLP

Input: Initial point $w^{(0)} \in \mathcal{F}$, projection matrix P, initial trust-region radius $\Delta^{(0)} \in (0, \tilde{\Delta}]$, feasibility tolerance τ, convergence tolerance $\sigma_{\text{outer}} \in (0, 10^{-5})$

```
success ← false
for k = 0, 1, 2, ... do
    w̄^(k) ← solve (2)
    if |m_f^(k)(w̄^(k))| ≤ σ_outer then
        break
    (ŵ^(k), success) ←
      FeasIterations(w^(k), w̄^(k), Δ^(k), τ)
    if success = true then
        Compute ρ^(k) according to (3)
        Δ^(k+1) ← ΔUpdate(ρ^(k), w^(k), w̄^(k), Δ^(k))
        (w^(k+1), −) ← Acceptance(ρ^(k), w^(k), ŵ^(k))
    else
        Δ^(k+1) ← α1 ||P(w̄^(k) − w^(k))||∞
        w^(k+1) ← w^(k)
return w^(k)
```

decides upon step acceptance, where

$$\Delta f^{(k)} = f(w^{(k)}) - f(\hat{w}^{(k)}), \; \Delta m_f^{(k)} = -m_f^{(k)}(\bar{w}^{(k)}). \tag{4}$$

The objective serves as merit function since all iterates remain feasible. Note that ρ has the subscript II for consistency with the extension discussed in Section IV.

Algorithm 2: Trust-region radius update

Parameter: $\tilde{\Delta} \geq 1$, $\alpha_1 \in (0,1)$, $\alpha_2 \in (1, \infty)$, $0 < \eta_1 < \eta_2 < 1$

```
Procedure ΔUpdate(ρ, w, w̄, Δ)
    if ρ < η1 then
        return α1 ||P(w̄ − w)||∞
    else if ρ > η2 and ||P(w̄ − w)||∞ = Δ then
        return min(α2 Δ, Δ̃)
    else
        return Δ
```

B. Feasibility Iterations

The feasibility iterations solve a sequence of parametric LPs. In order to distinguish between outer and inner iterations, a second superscript is introduced: $w^{(k,l)}$ denotes the iterate at the k-th outer iteration and at the l-th inner iteration.

Given $\bar{w}^{(k)}$ the solution of (2), the initial iterate $w^{(k,0)}$ for the feasibility iterations is set to $\bar{w}^{(k)}$ and the parametric

Algorithm 3: Acceptance test of the trial iterate

Parameter: $\sigma \in (0, 1/4)$
Procedure $\mathrm{Acceptance}(\rho, w, \hat{w})$
  if $\rho > \sigma$ **then**
    return $(\hat{w}, \mathrm{true})$
  else
    return (w, false)

linear problem $\mathrm{PLP}(w^{(k,l)}; w^{(k)}, \Delta^{(k)})$ is defined by:

$$\begin{aligned} \min_{w \in \mathbb{R}^{n_w}} \quad & (\nabla f^{(k)})^\top w \\ \text{s.t.} \quad & g(w^{(k,l)}) + J_g^{(k)}(w - w^{(k,l)}) = 0, \\ & h(w^{(k,l)}) + J_h^{(k)}(w - w^{(k,l)}) \leq 0, \\ & ||P(w - w^{(k)})||_\infty \leq \Delta^{(k)}. \end{aligned} \tag{5}$$

Problem (5) is similar to (2), but it is centered at $w^{(k,l)}$ and the constraints are evaluated at $w^{(k,l)}$. Consequently, the feasibility iterations are relatively cheap since only constraint evaluations (no derivatives) are required.

The optimal solution of (5) is $w^{(k,l+1)} := w^*_{\mathrm{PLP}}(w^{(k,l)}; w^{(k)}, \Delta)$. The algorithm continues until an iterate is feasible and fulfills the projection ratio condition, for details see [7]. A heuristic checks whether such an iterate is likely to be found. If this is not the case, the feasibility iterations failed, which results in a decrease of the trust-region radius in Algorithm 1. Only a linear convergence rate is expected from the algorithm.

Algorithm 4: Feasibility Iterations

Procedure $\mathrm{FeasIterations}(w^{(k)}, \bar{w}^{(k)}, \Delta, \tau)$
  $w^{(k,0)} \leftarrow \bar{w}^{(k)}$, success $\leftarrow$ false
  for $l = 0, 1, 2, \ldots$ **do**
    if $v(w^{(k,l)}) \leq \tau$ and $\|\bar{w}^{(k)} - w^{(k,l)}\| / \|\bar{w}^{(k)} - w^{(k)}\| < 1/2$ **then**
      success $\leftarrow$ true, **break**
    else if $w^{(k,l)}$ *diverges according to [7]* **then**
      break
    $w^{(k,l+1)} \leftarrow$ solve $\mathrm{PLP}(w^{(k,l)}, w^{(k)}, \Delta)$
  return $(w^{(k,l)}, \mathrm{success})$

III. Practical Limitations of FSLP

Several practical deficiencies of FSLP, including the feasibility iterations and the initialization at a feasible point, were identified during extensive testing.

A. Feasibility Iterations

The limitations of the feasibility iterations are illustrated with two examples. The first section addresses the issue of infeasible subproblems due to incompatible constraints. The second problem shows the convergence of the feasibility iterations only for small trust-region radii. The consequence of these drawbacks is that the FSLP algorithm often takes small steps while guaranteeing the feasibility of every iterate, which causes many additional constraint evaluations and the solution of many additional LPs.

1) Infeasible Subproblems: Even though the constraints were linearized at a feasible point, the subproblems in subsequent feasibility iterations may be infeasible. This is illustrated by the test problem:

$$\min_{w \in \mathbb{R}^2} w_2 \quad \text{s.t.} \quad w_2 \geq w_1^2, w_2 \geq 0.1 w_1. \tag{6}$$

Its solution is $w^* = (0, 0)$, as shown in Figure 1. Infeasible areas are grayed out and the solution is shown as a diamond.

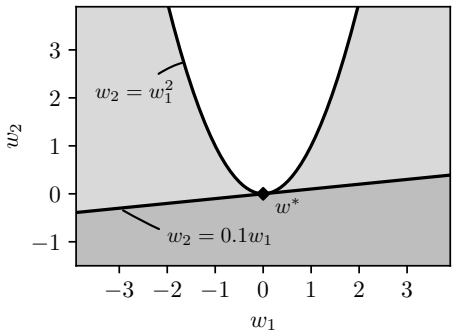


Fig. 1: Visual representation of Example (6).

Since the objective and the second constraint are linear, only the linearization of the first constraint varies over the iterations. Similarly, the trust-region radius is fixed throughout feasibility iterations. The FSLP algorithm is started from a point $w^{(0)} = (1, 3)$ with an initial trust-region radius $\Delta^{(0)} = 4$. The LP of the first outer iteration is:

$$\min_{w \in \mathbb{R}^2} \quad w_2 \tag{7a}$$
$$\text{s.t.} \quad w_2 \geq -1 + 2w_1, \tag{7b}$$
$$w_2 \geq 0.1 w_1, \tag{7c}$$
$$-3 \leq w_1 \leq 5, -1 \leq w_2 \leq 7. \tag{7d}$$

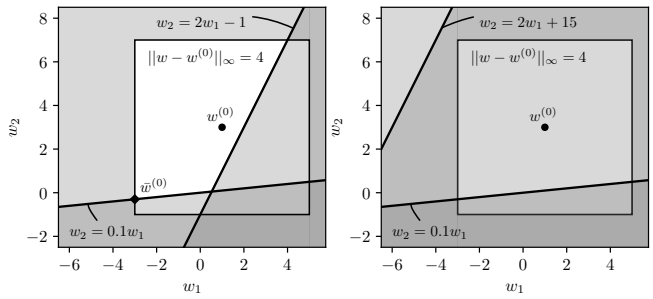
Its solution is $(-3, -0.3)$, as shown in Figure 2a. Infeasible areas are grayed out and the solution is shown as a diamond. The linearized constraint of the first feasibility iteration is:

$$w_2 \geq 15 + 2w_1, \tag{8}$$

which is a parallel displacement of (7b). Figure 2b illustrates that (8) is shifted outside of the trust region, i.e., the LP becomes infeasible.

We note that this also holds for Anderson Accelerated FSLP since the infeasibility occurs within the first iteration of the feasibility iterations, which is the first iteration for the Anderson acceleration.

2) Convergence of Feasibility Iterations for Small Trust-Region Radii: In this section, we analyze the maximum trust-region radius such that the feasibility iterations converge towards a feasible point, while maintaining feasible subproblems and without reaching the maximum number of



(a) First outer iteration. (b) First feasibility iteration.

Fig. 2: Illustration of Feasibility Iterations leading to an infeasible subproblem. Grayed-out areas are infeasible.

feasibility iterations. For $n \geq 2$, we are interested in the following high-dimensional nonlinear problem:

$$\min_{w \in \mathbb{R}^n} \quad -w_1 \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i^2 - 1 = 0,$$

that is finding the maximum w_1 on an n-dimensional unit sphere. The feasible initial guess is chosen as $w^{(0)} = (0.5, \sqrt{1 - 0.5^2}, 0, \ldots, 0)$, i.e., it lies on the two-dimensional unit sphere. In our implementation, $n \in \{2, 3, 4, \ldots, 5000\}$, the feasibility tolerance τ is set to 10^{-8} and the maximum number of feasibility iterations is set to 100. The experiment is started with a trust-region radius of 10 and it is always halved in case the feasibility iterations do not converge. Figure 3 illustrates the decrease of the maximum trust-region radius that guarantees convergence with the dimension n.

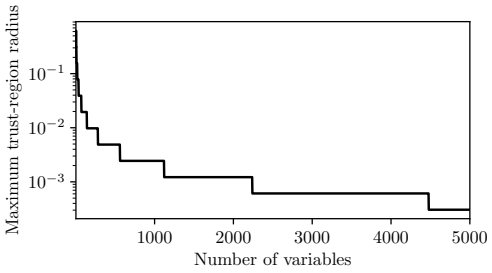


Fig. 3: Maximum trust-region radii for convergence of feasibility iterations with respect to the number of variables.

This trend was also observed when solving time-optimal control problems (TOCP) with FSLP. The dimensions of the TOCPs typically range from 100 to several $1,000$.

B. Feasible Initialization

In the general case of (1), it is arduous to initialize the problem at a feasible point. There are two straightforward techniques to obtain a feasible initial guess: i) a feasibility phase that minimizes constraint violation and (hopefully) returns a feasible initial guess [8], [11]; or ii) an ℓ_1 relaxation of the problem (dropping the equality constraints for simplicity):

$$\min_{w \in \mathbb{R}^{n_w}} \quad f(w) + \mu e^\top s \quad \text{s.t.} \quad g(w) \leq s,\ s \geq 0,$$

where e is a vector of ones of appropriate size and $\mu > 0$. For a given w, feasibility is achieved by setting the elastic variables s to sufficiently large values. It is well known that the ℓ_1 relaxation is exact and that the relaxed problem has the same solution as the original problem (1) provided that the penalty parameter μ is large enough [9]. Moreover, a dynamic penalty parameter update is required for fast convergence.

In [6], [7], we decided to relax as few constraints as possible to stay as close as possible to the original problem. In particular, we were interested in solving time-optimal point-to-point motion problems, i.e., initial and terminal conditions are given for the state of the controlled system and the goal is to transfer the system from the initial to the terminal state in minimal time. The considered test problems described obstacle avoidance for one obstacle using an explicit Runge-Kutta 4 integrator for the dynamics. These problems were highly nonlinear and challenging from a mathematical programming perspective, but simple in terms of the system's environment that needed to be controlled. We easily found initial guesses that were feasible with respect to all nonlinear constraints, but not with respect to the initial and terminal conditions. Consequently, the algorithm was started at an almost feasible point. One question remains open: is it worth using the feasibility iterations to keep the problem feasible, even though the iterate is far from the original feasible set?

Further testing of FSLP on TOCPs allowed us to identify the following drawbacks with feasible initialization. In [7] and [6], only one obstacle was introduced in the problem. The situation gets more difficult when several obstacles are introduced or if the point-to-point motion is performed in corridors, or in a maze. If instead of an explicit integrator, we use an implicit integrator, like the implicit Euler method, feasible states are solutions of a system of nonlinear equations, which complicates the search for a feasible initial guess. The situation becomes even more challenging when a direct collocation approach is chosen, introducing additional collocation points between states. In this case, besides satisfying constraints at states and controls, states are interpolated by a polynomial between two consecutive states, and the dynamics constraints must be met at the collocation point. This complicates the initialization at a feasible point, leading to the necessity for additional slack variables. In this case, the relaxed problem resembles (III-B), signifying that the relaxed problem is feasible while the original problem is not.

C. Conclusion of Practical Limitations

From the previous sections, we conclude the following: First, the feasibility iterations converge towards a feasible point but only for small trust-region radii, which yields small steps and requires many additional constraint evaluations. In the implementation of FSLP, feasibility is defined with respect to a given tolerance τ typically set to 10^{-8}. One straightforward simplification is to relax this tolerance: the iterates are allowed to stay within a tolerance-tube (neigh-

borhood) around the feasible set. This has two advantages: the overall algorithm requires fewer feasibility iterations since feasibility can be achieved only loosely, and the algorithm can take larger steps since it is expected that the feasibility iterations will converge to the required accuracy with a larger trust-region radius.

Second, for hard TOCPs, initialization at a feasible point can become difficult and relaxing all the constraints does not seem to be the solution. Additionally, the user experience of FSLP is unfavorable since it is hard to initialize the algorithm. In order to avoid this situation, we propose to initialize the algorithm with a feasibility phase that allows infeasible initial guesses and brings the iterate into a neighborhood around the feasible set.

IV. An Almost Feasible SLP Algorithm

To overcome the issues mentioned in the previous section, we introduce a more general two-phase SLP algorithm, which we call almost feasible Sequential Linear Programming (afSLP), inspired by the tolerance-tube method described in [15]. A tolerance-tube (Figure 4) is a relaxation of the feasible set $\mathcal{F}$ parameterized by its width $\tau^{(k)} \in (0,1)$ and a parameter $\beta \in (0,1)$. β is used to guarantee that the tolerance-tube can be reduced while all the iterates stay inside the tolerance-tube.

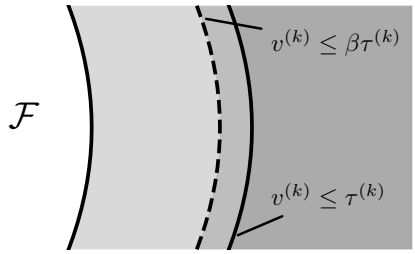


Fig. 4: Tolerance-tube around the feasible set $\mathcal{F}$. Grayed-out areas are infeasible.

Phase I improves feasibility until the tolerance-tube is reached. Phase II iterates towards optimality while keeping all iterates inside of the tolerance-tube. When a subproblem is infeasible, a feasibility restoration phase is invoked. Moreover, additional changes were made to the original algorithm that improved the global convergence behavior.

A. Phase I: Feasibility Phase

Let $w^{(k)}$ be outside of the tolerance-tube, i.e., $v^{(k)} > \beta\tau$. If the LP (2) is successfully solved, the linearized constraint violation for the solution $\bar{w}^{(k)}$ is zero. The actual and predicted reductions in constraint violation at $\bar{w}^{(k)}$ are

$$\Delta v^{(k)} = v^{(k)} - v(\bar{w}^{(k)}), \quad \Delta m_v^{(k)} = v^{(k)}$$

and the ratio of actual and predicted reduction is given by

$$\rho_{\mathrm{I}}^{(k)} := \Delta v^{(k)} / \Delta m_v^{(k)}.$$

The step is accepted or rejected and the trust-region radius is adjusted as described in the FSLP algorithm. The algorithm stays in Phase I until the iterates reach the tolerance-tube, i.e., $v^{(k)} \leq \beta\tau$, then switches to Phase II. If however (2) is infeasible, the algorithm switches to a feasibility restoration phase described in Section IV-C.

B. Phase II: Optimality Phase

Let $w^{(k)}$ be inside of the tolerance-tube, i.e., $v^{(k)} \leq \beta\tau$. If $v(\bar{w}^{(k)}) > \beta\tau$, where $\bar{w}^{(k)}$ is the solution of (2), the feasibility iterations (Algorithm 4) are invoked to bring the trial iterate back in the tolerance-tube. Here, we use an unconstrained sufficient decrease condition (only the objective is considered): the actual and predicted reductions and their quotient are defined as in the FSLP algorithm.

In constrained optimization, the predicted reduction may not be positive, since reducing infeasibility may increase the objective. To address this and to avoid infinitely small steps that converge towards an infeasible point, we use the switching condition traditionally used in filter methods [3]:

$$\Delta m_f^{(k)} \geq \sigma v^{(k)} \tag{9}$$

with $\sigma \in (0,1)$. If the switching condition is satisfied, the trust-region update strategy (Algorithm 2) is invoked. Otherwise, sufficient decrease does not seem to be possible, the step is rejected and the trust-region radius is decreased. afSLP terminates if a KKT point is found. If the subproblem is infeasible, the algorithm switches to the feasibility restoration phase.

C. Feasibility Restoration Phase

The feasibility problem is defined as:

$$\min_{w \in \mathbb{R}^{n_w}} \quad v_{\mathrm{R}}(w) := \|g(w)\|_1 + \|[h(w)]^+\|_1. \tag{10}$$

Due to the ℓ_1 norm, the problem is non-smooth, but by introduction of elastic variables, it can be reformulated as a smooth constrained problem. Let $t^+, t^- \in \mathbb{R}^{n_g}, s \in \mathbb{R}^{n_h}$, then (10) is equivalent to

$$\begin{aligned} \min_{w,s,t^+,t^-} \quad & \sum_{i=1}^{n_g}(t_i^+ + t_i^-) + \sum_{i=1}^{n_h} s_i \\ \text{s.t.} \quad & g(w) - t^+ + t^- = 0 \\ & h(w) - s \leq 0 \\ & s, t^+, t^- \geq 0. \end{aligned} \tag{11}$$

Applying sequential linear programming on (11), we obtain the trust-region LP:

$$\begin{aligned} \min_{w,s,t^+,t^-} \quad & \sum_{i=1}^{n_g}(t_i^+ + t_i^-) + \sum_{i=1}^{n_h} s_i \\ \text{s.t.} \quad & g^{(k)} + J_g^{(k)}(w - w^{(k)}) - t^+ + t^- = 0, \\ & h^{(k)} + J_h^{(k)}(w - w^{(k)}) - s \leq 0, \\ & s, t^+, t^- \geq 0, \\ & ||P(w - w^{(k)})||_\infty \leq \Delta^{(k)}. \end{aligned} \tag{12}$$

Note that elastic variables are excluded from the trust-region constraint so that the LP is feasible. Let $w_{\mathrm{R}}^{(k)}$ be the w

component of the solution of (12). We define the model of the ℓ_1 feasibility measure v_{R} by

$$m_{\mathrm{R}}^{(k)}(w) := \|g^{(k)} + J_g^{(k)}(w - w^{(k)})\|_1 + \|[h^{(k)} + J_h^{(k)}(w - w^{(k)})]^+\|_1,$$

the actual and predicted reductions by

$$\Delta v_{\mathrm{R}}^{(k)} = v_{\mathrm{R}}^{(k)} - v_{\mathrm{R}}(w_{\mathrm{R}}^{(k)}),$$
$$\Delta m_{\mathrm{R}}^{(k)} = v_{\mathrm{R}}^{(k)} - m_{\mathrm{R}}^{(k)}(w_{\mathrm{R}}^{(k)})$$

and their ratio by

$$\rho_{\mathrm{R}}^{(k)} := \Delta v_{\mathrm{R}}^{(k)} / \Delta m_{\mathrm{R}}^{(k)}.$$

Since (10) is unconstrained, the predicted reduction is always non-negative and no switching condition is necessary.

The acceptance test (Algorithm 3) decides upon acceptance or rejection of the trial iterate. The main difference in our approach compared to [15] is that τ is adaptively updated and denoted by $\tau^{(k)}$ instead. If $v^{(k)} < \beta\tau^{(k)}$ and the algorithm switches to feasibility restoration, then we require that $v(\bar{w}^{(k)}) < \beta\tau^{(k)}$. If the trial iterate is accepted, the tolerance-tube width is reduced: $\tau^{(k+1)} \leftarrow \beta\tau^{(k)}$ (line 27 in Algorithm 5). This is a safeguard that prevents the algorithm from cycling. The parameter β guarantees that the tolerance-tube can be reduced while keeping the current iterate inside.

The algorithm terminates when a small step is encountered. If the current iterate is infeasible, this is an indication that the problem might be infeasible.

D. The Complete Algorithm

The full afSLP method is described in Algorithm 5. Since feasibility is not maintained at every iteration, we require for termination that:

$$\begin{cases} v(w^*) \le \varepsilon_{\mathrm{F}} & \text{(feasibility)} \\ m_f(w^*, \bar{w}^*) \le \varepsilon_{\mathrm{O}} & \text{(stationarity)} \end{cases}$$

for a given feasibility tolerance $\varepsilon_{\mathrm{F}} \in (0,1)$ and optimality tolerance $\varepsilon_{\mathrm{O}} \in (0,1)$. If the trust-region radius is decreased below a minimum value, the algorithm terminates with an error message.

Global convergence could probably be proved in a similar fashion to [3]. We note that unlike afSLP, the original algorithm does not possess a switching condition (9), it does not decrease the tolerance-tube when the feasibility restoration is invoked in Phase II, and it performs a second-order correction, i.e., only one iteration of feasibility iterations, if the trial iterate after solving (2) is outside the tolerance-tube. We now describe an example where the original tolerance-tube algorithm [15] would cycle:

$$\min_{w \in \mathbb{R}^2} w_2 \quad \text{s.t.} \quad w_2 \ge w_1^2 + 0.0375, w_1 \ge w_2. \tag{13}$$

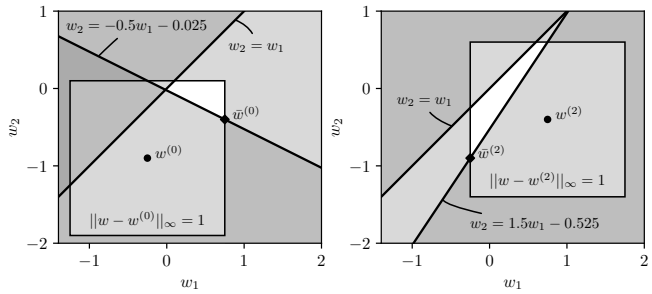
This example is similar to (6), except that the quadratic constraint is shifted by 0.0375 so as to obtain nice iterates in the cycle, and the inequality is flipped in the second constraint while removing the factor 0.1. Without loss of generality, we set the tolerance-tube width to $\tau = 1$ (for lower values of τ, an equivalent problem can be obtained by scaling the constraints – therefore the constraint violation – by τ). The initial trust-region radius is set to $\Delta^{(0)} = 1$.

We start the original tolerance-tube method at $w^{(0)} = (-0.25, -0.9)$. Since $v^{(0)} = 1$, the initial point lies on the tolerance-tube, therefore the algorithm starts in Phase II. The feasible set of the initial LP is illustrated in Figure 5a and its solution is $\bar{w}^{(0)} = (0.75, -0.4)$. The trial iterate is accepted and the trust-region radius is increased: $(w^{(1)}, \Delta^{(1)}) = (\bar{w}^{(0)}, 2)$. The solution of the next LP and the second-order corrected iterate lie outside of the tolerance-tube. Therefore, the trial iterate is rejected and the trust-region radius is decreased: $(w^{(2)}, \Delta^{(2)}) = (w^{(1)}, 1)$. The feasible set of

Algorithm 5: afSLP

Input: Initial point $w^{(0)}$, projection matrix P, initial trust-region radius $\Delta^{(0)} \in (0, \tilde{\Delta}]$, initial tolerance-tube $\tau^{(0)}$, $\beta, \sigma, \varepsilon_{\mathrm{F}}, \varepsilon_{\mathrm{O}} \in (0,1)$

1 **for** $k = 0, 1, 2, \ldots$ **do**
2 $\bar{w}^{(k)} \leftarrow$ solve (2)
3 **if** (2) *is feasible* **then**
4 update_radius $\leftarrow$ false
5 **if** $v^{(k)} > \beta\tau^{(k)}$ **then** `// feasibility phase`
6 update_radius $\leftarrow$ true; $\rho^{(k)} \leftarrow \rho_{\mathrm{I}}^{(k)}$
7 **else** `// optimality phase`
8 **if** $v^{(k)} \le \varepsilon_{\mathrm{F}}$ *and* $m_f^{(k)}(\bar{w}^{(k)}) \le \varepsilon_{\mathrm{O}}$ **then**
9 **break**
10 **if** $v(\bar{w}^{(k)}) \le \beta\tau^{(k)}$ **then**
11 success $\leftarrow$ true; $\hat{w}^{(k)} \leftarrow \bar{w}^{(k)}$
12 **else**
13 $(\hat{w}^{(k)}, \text{success}) \leftarrow$ FeasIterations$(w^{(k)}, \bar{w}^{(k)}, \Delta^{(k)}, \tau^{(k)})$
14 **if** success *and* $\Delta m_f^{(k)} \ge \sigma v^{(k)}$ **then**
15 update_radius $\leftarrow$ true; $\rho^{(k)} \leftarrow \rho_{\mathrm{II}}^{(k)}$
16 **if** update_radius **then**
17 $\Delta^{(k+1)} \leftarrow$ ΔUpdate$(\rho^{(k)}, w^{(k)}, \bar{w}^{(k)}, \Delta^{(k)})$
18 $(w^{(k+1)}, -) \leftarrow$ Acceptance$(\rho^{(k)}, w^{(k)}, \hat{w}^{(k)})$
19 **else**
20 $\Delta^{(k+1)} \leftarrow \alpha_1 ||P(\bar{w}^{(k)} - w^{(k)})||_\infty$
21 $w^{(k+1)} \leftarrow w^{(k)}$
22 **else** `// restoration phase`
23 $\bar{w}^{(k)} \leftarrow$ solve (12); $\rho^{(k)} \leftarrow \rho_{\mathrm{R}}^{(k)}$
24 $\Delta^{(k+1)} \leftarrow \Delta\text{Update}(\rho^{(k)}, w^{(k)}, \bar{w}^{(k)}, \Delta^{(k)})$
25 $(w^{(k+1)}, \text{accept}) \leftarrow$ Acceptance$(\rho^{(k)}, w^{(k)}, \bar{w}^{(k)})$
26 **if** $v^{(k)} \le \beta\tau^{(k)}$ *and* accept **then**
27 $\tau^{(k+1)} \leftarrow \beta\tau^{(k)}$
28 **return** $w^{(k)}$



(a) First iteration. (b) Second iteration.

Fig. 5: Cycling in the original tolerance-tube algorithm.

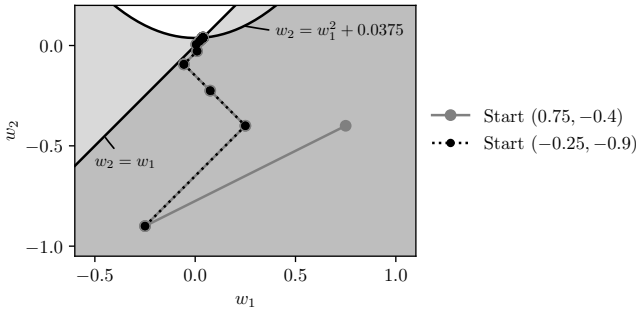


Fig. 6: Convergence of afSLP algorithm on Problem (13). Grayed-out areas are infeasible.

this LP is shown in Figure 5b and its solution is $\bar{w}^{(2)} = (-0.25, -0.9)$. The trial iterate is accepted and the trust-region radius is again increased: $(w^{(3)}, \Delta^{(3)}) = (\bar{w}^{(2)}, 2)$. At this point, the original tolerance-tube algorithm starts cycling.

The predicted reduction reveals that $\Delta m_f^{(0)}(\bar{x}^{(0)}) = -0.5$ and $\Delta m_f^{(2)}(\bar{x}^{(0)}) = 0.5$, but $\rho^{(0)} = \rho^{(2)} = 1$: even though the algorithm does not predict a decrease of the objective in the first LP, the trial iterate is accepted since $\rho^{(k)}$ is always 1. The switching condition (9) introduced in afSLP prevents the algorithm from taking these steps.

V. Simulation Results

This section presents simulation results that demonstrate the effectiveness of afSLP on two problems: (i) the test problem in (13), and (ii) the time-optimal point-to-point motion (in the Euclidean space) of a parallel SCARA robot.

The OCPs considered in this simulation are implemented using IMPACT [4], a framework for rapid prototyping of nonlinear model predictive control built on top of CasADi which inherits the solvers implemented therein. All simulations were carried out using an Intel core i7-10810U CPU and the LP solver CPLEX version 12.8 [5]. The algorithmic parameters are chosen as in [6].

A. Test problem (13)

We demonstrate that, unlike the original tolerance-tube algorithm [15], afSLP achieves convergence on Problem (13). Since the original tolerance-tube algorithm only checks for $v^{(k)} \leq \tau$ with $\tau = 1$ in Problem (13), we choose $\tau^{(0)} = 1.2$ and $\beta = 0.9$ to have a similar maximum allowed constraint violation in the initial iteration. The algorithm is started from both cycling points $(-0.25, -0.9)$ and $(0.75, -0.4)$. Figure 6 shows the convergence of afSLP towards the optimal solution $w^* \approx (0.039, 0.039)$ for both initial guesses. As in the analytical example, starting at $(0.75, -0.4)$ brings the next iterate to $(-0.25, -0.9)$, then the iterations for both starting points coincide.

B. Time-optimal motion of a parallel SCARA robot

To further demonstrate the effectiveness of afSLP, we apply it to the TOCP for the SCARA motion example originally introduced in our previous work [6]. For the sake of conciseness, only a brief description in the original notation is given below.

The parallel SCARA robot consists of two 2-link arms whose end effectors are attached to a revolute joint j_5 (loop closure constraint). It has two actuated joints (j_1 and j_3) and three unactuated joints (j_2, j_4 and j_5). Let us denote the configuration of the i-th joint as $q_i \in \mathbb{S}^1$, the independent coordinates as $q := [q_1, q_3]^\top$, the generalized coordinates as $\bar{q} := \sigma(q) = [q_1, q_2(q), q_3, q_4(q)]^\top$, the inertia matrix as M and the vector of Coriolis and centrifugal effects as F. For a state vector $x := [q^\top, \dot{q}^\top]^\top$ and a control (torque) input vector $u := \tau_\mathrm{S} \in \mathbb{R}^2$, the dynamics of the robot [2] are described by

$$\dot{x} = f_\mathrm{ode}(x, u) := [\dot{q}^\top,\ (M^{-1}(\bar{q})(\tau_\mathrm{S} - F(\bar{q}, \dot{\bar{q}})))^\top]^\top.$$

It is discretized for a sampling time h_t using a 4th-order Runge-Kutta integrator, which results in a function $f(x_k, u_k, h_t)$. The TOCP for the SCARA motion example is formulated as

$$\min_{\substack{x_0,\dots,x_N \\ u_0,\dots,u_{N-1} \\ s_0,s_N,T}} \ T + \mu_0^\top s_0 + \mu_N^\top s_N \tag{14a}$$

$$\text{s.t.} \quad -s_0 \leq x_0 - \overline{x}_0 \leq s_0, \tag{14b}$$

$$x_{k+1} = f(x_k, u_k, h_t), \quad 0 \leq k \leq N-1, \tag{14c}$$

$$u_k \in \mathbb{U}, \quad 0 \leq k \leq N-1, \tag{14d}$$

$$x_k \in \mathbb{X}, \quad 0 \leq k \leq N, \tag{14e}$$

$$e(x_k, u_k) \leq 0, \quad 0 \leq k \leq N-1, \tag{14f}$$

$$-s_N \leq x_N - \overline{x}_N \leq s_N, \tag{14g}$$

where $N \in \mathbb{N}$ is the horizon length, $T \in \mathbb{R}_{>0}$ is the time horizon, $h_t := T/N$, $x_k \in \mathbb{R}^{n_x}, u_k \in \mathbb{R}^{n_u}$ are the state and control variables, $\bar{x}_0, \bar{x}_N \in \mathbb{R}^{n_x}$ are the initial and terminal states, $\mu_0, \mu_N \in \mathbb{R}^{n_s}_{>0}$ are penalty parameters for the elastic variables $s_0, s_N \in \mathbb{R}^{n_x}$, and the sets $\mathbb{U}, \mathbb{X}$ are convex polytopes defined by lower and upper bounds on τ, q and $\dot{q}$. Finally, the function $e(x_k, u_k)$ encloses task-related stage constraints: (i) an upper bound V_max on the squared ℓ_2 norm of the velocity of the end-effector $\dot{p}_\mathrm{ee}$ ($\|\dot{p}_\mathrm{ee}\|^2 \leq V_\mathrm{max}^2$), and (ii) a collision avoidance constraint based on separating hyperplanes:

$$\mathbf{n}_\mathrm{a}^\top p_\mathrm{ee} + \mathbf{n}_\mathrm{b} + r_\mathrm{s} \leq 0,\ \mathbf{n}_\mathrm{a}^\top v + \mathbf{n}_\mathrm{b} \geq 0,\ \forall v \in \mathcal{V}_\mathrm{obs}, \tag{15}$$

where $\mathbf{n} := [\mathbf{n}_\mathrm{a}^\top, \mathbf{n}_\mathrm{b}]^\top \in \{\mathbf{v} \in \mathbb{R}^3 : \|\mathbf{v}\|_\infty \leq 1\}$ defines the hyperplane, $r_\mathrm{s} \in \mathbb{R}_{\geq 0}$ is a safety margin, and $\mathcal{V}_\mathrm{obs}$ represents the vertices of an obstacle.

We compare afSLP against the original FSLP algorithm [6] and the state-of-the-art nonlinear optimization solver IPOPT [12] for a variation of problem (14) in which the initial and terminal constraints (14b) and (14g) are not relaxed.

1) Comparison against FSLP: Table I shows how the performance (number of constraint evaluations and average wall time for 100 runs of the problem) of FSLP and afSLP varies for different values of $\tau^{(0)}$. The number of constraint evaluations and the average wall time decrease for increasing values of $\tau^{(0)}$ until $\tau^{(0)} = 10^{-4}$, above which the performance decreases. Overall, afSLP reduces the wall time by 50%.

TABLE I: Number of constraint evaluations and average wall time and on the SCARA test problem.

	FSLP	afSLP				
$\tau^{(0)}$	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
# g, h eval.	723	612	502	557	316	268
wall time (s)	0.576	0.447	0.361	0.423	0.283	0.287

2) Comparison against IPOPT without constraint relaxation: afSLP is compared against IPOPT on the SCARA test problem for different time horizons N. We choose $\tau^{(0)} = 10^{-3}$, $\beta = 0.9$, $\varepsilon_\mathrm{O} = \varepsilon_\mathrm{F} = 10^{-7}$. The optimality and feasibility tolerances of IPOPT are set to 10^{-7}. The maximum iteration number for both solvers was set to $1,000$. For every N the experiment was run 100 times. Figure 7 shows the average wall times for IPOPT and afSLP. afSLP is significantly faster than IPOPT, which is also due to the fact that expensive evaluations of the Hessian matrix are not required.

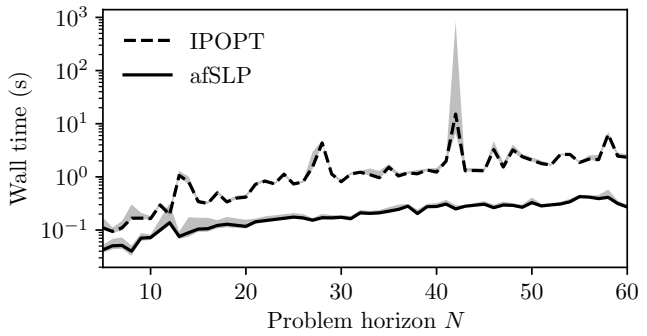


Fig. 7: Comparison of average wall time of IPOPT and afSLP for different problem horizons N.

VI. Conclusion

In this paper, we proposed illustrative examples that exhibit the practical deficiencies of FSLP, a feasible sequential linear programming method for solving TOCPs. We then introduced afSLP, an almost feasible SLP method that alleviates the FSLP shortcomings: it allows infeasible initial guesses, iterates first towards the feasible set until a tolerance-tube is reached, then iterates towards optimality. It is inspired by a tolerance-tube method introduced in [15]. We demonstrated in a simple example the lack of global convergence of the original tolerance-tube method and proposed two enhancements that overcome these limitations. The performance of afSLP was assessed on a SCARA robot optimization problem and compared against FSLP and the state-of-the-art nonlinear optimization solver IPOPT.
Future work will focus on establishing a global convergence proof, on improving the local convergence behavior and on providing suboptimal feasibility guarantees even though afSLP maintains a relaxed feasible set.

References

[1] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – a software framework for nonlinear optimization and optimal control. *Math. Program. Comput.*, 11(1):1–36, 2019.

[2] L. Cheng, Y. Lin, Z.-G. Hou, M. Tan, J. Huang, and W. J. Zhang. Adaptive tracking control of hybrid machines: A closed-chain five-bar mechanism case. *IEEE/ASME Trans. Mechatron.*, 16(6):1155–1163, dec 2011.

[3] R. Fletcher, S. Leyffer, and P. Toint. On the global convergence of an SLP-filter algorithm. *Numerical Analysis Report NA/183, University of Dundee, UK*, 12 1999.

[4] A. Florez, A. Astudillo, W. Decré, J. Swevers, and J. Gillis. IMPACT: A toolchain for nonlinear model predictive control specification, prototyping, and deployment. In *Proceedings of the IFAC World Congress*, Yokohama, Japan, 2023.

[5] Cplex IBM ILOG. V12.8: User's manual for CPLEX, 2017.

[6] D. Kiessling, P. Pas, A. Astudillo, P. Patrinos, and J. Swevers. Anderson accelerated feasible sequential linear programming. In *Proceedings of the IFAC World Congress*, Yokohama, Japan, 2023.

[7] D. Kiessling, A. Zanelli, A. Nurkanović, J. Gillis, M. Diehl, M. Zeilinger, G. Pipeleers, and J. Swevers. A feasible sequential linear programming algorithm with application to time-optimal path planning problems. In *Proceedings of 61st IEEE Conference on Decision and Control*, Cancun, Mexico, December 2022.

[8] L. Nita and E. Carrigan. SEQUOIA: A sequential algorithm providing feasibility guarantees for constrained optimization. In *Proceedings of the IFAC World Congress*, Yokohama, Japan, 2023.

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

[10] M. J. Tenny, S. J. Wright, and J.B. Rawlings. Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming. *Comput. Optim. Appl.*, 28:87–121, 2004.

[11] A. Tits and C. Lawrence. Nonlinear equality constraints in feasible sequential quadratic programming. *Optimization Methods and Software*, 6, 02 1997.

[12] A. Wächter and L. 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.

[13] S. J. Wright and M. J. Tenny. A feasible trust-region sequential quadratic programming algorithm. *SIAM Journal on Optimization*, 14:1074–1105, 1 2004.

[14] J. L. Zhou and A. Tits. User's guide for FSQP version 3.0c: A FORTRAN code for solving constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality and linear constraints. 01 1992.

[15] C. Zoppke-Donaldson. *A tolerance-tube approach to sequential quadratic programming with applications*. PhD thesis, University of Dundee, 1995.