Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)

POLITECNICO
MILANO 1863

Constrained Numerical Optimization for Estimation and Control

Laboratory session G - Constrained continuous nonlinear programming via SQP: algorithm and application to Finite Horizon Optimal Control Problem

Course 052358 - M.Sc. Programme in Automation and Control Engineering

Date: September 12, 2021

Version: 1.0

Authors: Lorenzo Fagiano
Marco Lauricella

Contacts: Lorenzo Fagiano
+39.02.2399.9609
lorenzo.fagiano@polimi.it

Contents

1 Introduction and learning goals

In this laboratory session we develop and test numerical optimization algorithms for constrained, continuous nonlinear programs (NLPs) of the form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \tag{1a}$$

s.t.

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1b}$$

$$C\boldsymbol{x} \geq \boldsymbol{d} \tag{1c}$$

$$g(\boldsymbol{x}) = 0 \tag{1d}$$

$$h(\boldsymbol{x}) \geq 0 \tag{1e}$$

This is the most general class of problems treated in this course, and it appears in many estimation and control tasks, such as parameter identification of nonlinear models and/or with a simulation error criterion and constraints on the parameters to be identified; state/disturbance/parameter estimation in presence of a nonlinear model and constraints on the input and/or state variables; constrained nonlinear optimal control over a finite horizon. The reason why linear equality and inequality constraints (1b)-(1c) are included separately is related to computational efficiency: in fact, for these constraints the gradients are directly available (matrices A^T and C^T) and need not be computed numerically.
The algorithms developed in this session implement the so-called Sequential Quadratic Programming methods, in particular with two options to compute the Hessian of the quadratic sub-problem: constrained Gauss-Newton and BFGS. These algorithm need the following main building blocks:

1. Gradient computation (we will use the algorithms developed in Laboratory session D);
2. Back-tracking line search algorithm (we will use the algorithm developed in Laboratory session E);
3. A QP solver (we will use Matlab's `quadprog`);
4. An overall SQP algorithm embedding the QP and the line search with merit function.

At each iteration of the SQP algorithm, a QP is solved to obtain search directions in the space of primal variables (i.e. the decision variable $\boldsymbol{x}$) and of the Lagrange multipliers ($\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, respectively for equality and inequality constraints), see Chapter 6 of the course lecture notes. After the QP has been solved, the line search is carried out by using a merit function and its directional derivative, that can be both computed based on the local information (cost, constraints, and their gradients) available at each iteration.
The goal of this session is to code a function (e.g. `myfmincon`) which will be the main solver in your constrained NLP suite. Such a function will realize item 4. in the list above.
To test the optimization routine, two cases will be used: an academic test function with many local minima and different types of constraints, and the cost function resulting from a finite horizon optimal control problem where the goal is to achieve the fastest average speed of a vehicle on a double-turn path, with constraints on both inputs and states.
The learning goals are:

- To learn how to code a numerical optimization routine based on Sequential Quadratic Programming;

- To learn how to use the developed routine to solve continuous NLPs, and to evaluate the effectiveness of the different approaches.

2 Sequential Quadratic Programming

For convenience, a typical SQP algorithm is reported in this section. Before the algorithm, we also recall the ℓ_1-norm merit function:

$$T_1(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \sigma_i |g_i(\boldsymbol{x})| + \sum_{i=1}^{q} \tau_i \max(0, -h_i(\boldsymbol{x})). \tag{2}$$

and its directional derivative, already simplified considering that $\boldsymbol{p}^{k*}$ is the solution to a QP featuring the linearized equality and inequality constraints:

$$D(T_1(\boldsymbol{x}^k), \boldsymbol{p}^{k*}) = \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^k)^T \boldsymbol{p}^{k*} - \sum_{i=1}^{p} \sigma_i |g_i(\boldsymbol{x}^k)| - \sum_{i:h_i(\boldsymbol{x}^k)<0}^{q} \tau_i \nabla_{\boldsymbol{x}} h_i(\boldsymbol{x}^k)^T \boldsymbol{p}^{k*} \tag{3}$$

Constrained continuous NLP solver with SQP method.

- **Inputs**: cost function $f : \mathbb{R}^n \to \mathbb{R}$ (and $F(\boldsymbol{x})$ for Gauss-Newton method, see below), equality constraints $g : \mathbb{R}^n \to \mathbb{R}^p$, inequality constraints $h : \mathbb{R}^n \to \mathbb{R}^q$, initial guess $\boldsymbol{x}^0 \in \mathbb{R}^n$, initial values of σ_i, τ_j for the ℓ_1-norm merit function, $i = 1, \dots, p,\ \ j = 1, \dots, q$.
- **Parameters**: termination tolerances TOL_∇, $TOL_{\boldsymbol{x}}$, $TOL_f \in (0, 1)$, $TOL_{\text{constr}} > 0$ maximum number of iterations $N_{\max} \in \mathbb{N}$; line-search sub-routine parameters, additional parameters according to the employed Hessian approximation method (e.g. Gauss-Newton, BFGS).
- **Initialization**: $\Delta\boldsymbol{x}^0 = 1$, $\Delta f^0 = 1$, $\boldsymbol{\lambda}^0 = 0$, $\boldsymbol{\mu}^0 = 0$.
- **Algorithm**:
  1. Compute $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^k)$, $\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^k)$, $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}^k)$ with a differentiation method of choice (for BFGS, these are already available except for iteration $k = 0$, see item 7. below);
  2. Check the terminating conditions:
     a) if $|\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)^T \boldsymbol{p}^k| \le TOL_\nabla$ **or** $\Delta\boldsymbol{x}^k \le TOL_{\boldsymbol{x}}$ **or** $\Delta f^k \le TOL_f$ **or** $k \ge N_{\max}$ **and** $\|g(\boldsymbol{x}^k)\|_\infty \le TOL_{\text{constr}}$ **and** $\min(h(\boldsymbol{x}^k)) \ge TOL_{\text{constr}}$ **then exit**.
     In this case, either $\boldsymbol{x}^* = \boldsymbol{x}^k$ is a possible local minimizer (and $f(\boldsymbol{x}^*)$ a local minimum), or the algorithm has ended prematurely but still returns a feasible point.
     b) **if** $k \ge N_{\max}$ **and** $\left(\|g(\boldsymbol{x}^k)\|_\infty > TOL_{\text{constr}}$ **or** $\min h(\boldsymbol{x}^k) < TOL_{\text{constr}}\right)$ **then exit**.
     In this case, the algorithm has ended prematurely at an unfeasible point, or possibly the problem is not feasible;
  3. Compute the QP Hessian H^k as:
     – **Constrained Gauss-Newton**: $H^k = 2\nabla_{\boldsymbol{x}} F(\boldsymbol{x}^k) \nabla_{\boldsymbol{x}} F(\boldsymbol{x}^k)^T$, where it is assumed that $f(\boldsymbol{x}) = F(\boldsymbol{x})^T F(\boldsymbol{x})$ and $F : \mathbb{R}^n \to \mathbb{R}^N$ is provided by the user. To ensure that H^k is strictly positive definite, one can also modify it by adding a small positive quantity σ (e.g. 10^{-14}) on the diagonal;
     – **BFGS**: H^k computed with the BFGS update rule (see item 7. below), where $H^0 = I$;
  4. Solve the QP:

$$\min_{\boldsymbol{p}^k} \ \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^k)^T \boldsymbol{p}^k + \frac{1}{2} \boldsymbol{p}^{kT} H^k \boldsymbol{p}^k$$

$$\text{s.t.}$$

$$\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^k)^T \boldsymbol{p}^k + g(\boldsymbol{x}^k) = 0$$

$$\nabla_{\boldsymbol{x}} h(\boldsymbol{x}^k)^T \boldsymbol{p}^k + h(\boldsymbol{x}^k) \ge 0$$

Denote with $\boldsymbol{p}^{k*}$, $\tilde{\boldsymbol{\lambda}}^{k*}$, $\tilde{\boldsymbol{\mu}}^{k*}$ the obtained solution and Lagrange multipliers, and compute

$$\begin{aligned}\Delta\boldsymbol{\lambda}^k &= (\tilde{\boldsymbol{\lambda}}^{k*} - \boldsymbol{\lambda}^k)\\ \Delta\boldsymbol{\mu}^k &= (\tilde{\boldsymbol{\mu}}^{k*} - \boldsymbol{\mu}^k)\end{aligned}$$

with $\boldsymbol{\lambda}^0$, $\boldsymbol{\mu}^0 = 0$.

5. Update the weights σ_i, τ_i for the merit function $T_1(\boldsymbol{x})$ (2) as:

$$\begin{aligned}\sigma_i &= \max\left(|\tilde{\boldsymbol{\lambda}}^{k*}|, \frac{\sigma_i^- + |\tilde{\boldsymbol{\lambda}}^{k*}|}{2}\right)\\ \tau_j &= \max\left(|\tilde{\boldsymbol{\mu}}^{k*}|, \frac{\tau_j^- + |\tilde{\boldsymbol{\mu}}^{k*}|}{2}\right)\end{aligned}$$

for all $i = 1, \ldots, p$ and all $j = 1, \ldots, q$, where σ_j^-, τ_j^- are the previous values;

6. Compute the directional derivative $D(T_1(\boldsymbol{x}^k), \boldsymbol{p}^{k*})$ as in (3), and carry out the back-tracking line search (see Chapter 4 in the lecture notes and/or laboratory session E) using the merit function $T_1(\boldsymbol{x})$ (2) and directional derivative $D(T_1(\boldsymbol{x}^k), \boldsymbol{p}^{k*})$ to compute t^k. Then update:

$$\begin{aligned}\boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + t^k\boldsymbol{p}^{k*}\\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + t^k\Delta\boldsymbol{\lambda}^k\\ \boldsymbol{\mu}^{k+1} &= \boldsymbol{\mu}^k + t^k\Delta\boldsymbol{\mu}^k\end{aligned}$$

7. Only for BFGS:
   a) compute the gradients $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{k+1})$, $\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1})$, $\nabla_{\boldsymbol{x}} h(\boldsymbol{x}^{k+1})$ and:

$$\begin{aligned}\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1}, \boldsymbol{\mu}^{k+1}) &= \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{k+1}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1})\boldsymbol{\lambda}^{k+1} - \nabla_{\boldsymbol{x}} h(\boldsymbol{x}^{k+1})\boldsymbol{\mu}^{k+1}\\ \nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^{k}, \boldsymbol{\lambda}^{k+1}, \boldsymbol{\mu}^{k+1}) &= \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{k}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k})\boldsymbol{\lambda}^{k+1} - \nabla_{\boldsymbol{x}} h(\boldsymbol{x}^{k})\boldsymbol{\mu}^{k+1}\end{aligned}$$

   Then compute

$$\begin{aligned}\boldsymbol{y} &= \left(\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1}, \boldsymbol{\mu}^{k+1}) - \nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}^{k}, \boldsymbol{\lambda}^{k+1}, \boldsymbol{\mu}^{k+1})\right)\\ \boldsymbol{s} &= \left(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\right)\end{aligned}$$

   b) If $\boldsymbol{y}^T\boldsymbol{s} \leq \gamma\boldsymbol{s}^t H^k\boldsymbol{s}$ (where $\gamma \in (0, 1)$), then compute $\tilde{\boldsymbol{y}} = \boldsymbol{y} + \dfrac{\gamma\boldsymbol{s}^T H^k\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{y}}{\boldsymbol{s}^T H^k\boldsymbol{s} - \boldsymbol{s}^T\boldsymbol{y}}(H^k\boldsymbol{s} - \boldsymbol{y})$;
   c) Compute $H^{k+1} = H^k - \dfrac{H^k\boldsymbol{s}\boldsymbol{s}^T H^k}{\boldsymbol{s}^T H^k\boldsymbol{s}} + \dfrac{\boldsymbol{y}\boldsymbol{y}^T}{\boldsymbol{s}^T\boldsymbol{y}}$ (or using $\tilde{\boldsymbol{y}}$ instead of $\boldsymbol{y}$, see item 7.b));

8. update the relative changes of optimization variables and cost function, $\Delta\boldsymbol{x}^k$, Δf^k, as $\Delta\boldsymbol{x}^k = \dfrac{\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|}{\max(eps, \|\boldsymbol{x}^k\|)}$ and $\Delta f^k = \dfrac{|f(\boldsymbol{x}^{k+1}) - f(\boldsymbol{x}^k)|}{\max(eps, |f(\boldsymbol{x}^k)|)}$ where eps is a small quantity (e.g. 10^{-16}).

9. set $k = k + 1$, go to 1.

This algorithm either terminates with the value $\boldsymbol{x}^*$ of a possible local minimizer, or it stops due to too little progress (in either the relative change of local minimizer, or the relative improvement of the cost function) or too many iterations. In addition, the corresponding cost function value $f(\boldsymbol{x}^*)$ and the number of computed iterations k are available as outputs.

3 Laboratory session's assignments

This laboratory session is divided in four parts.

I) Prepare a function to initialize all parameters.
It is convenient to prepare a function (e.g. `myoptimset`) that creates a Matlab structure, e.g. `myoptions`, with default values for all of the involved parameters (differentiation methods, line search parameters, quasi-Newton method and parameters, etc.). For example, see the attached, commented Matlab code. The same function used for unconstrained optimization can be used and expanded for this purpose.

II) Code your own SQP routine.
At first, you will write (e.g. in Matlab) a function like this:
`[xstar,fxstar,k,exitflag,xsequence] = myfmincon(fun,x0,A,b,C,d,p,q,myoptions)`
as detailed in the attached, commented Matlab code. Note that we use a single cost function that will provide both the cost and the constraints, and one has to indicate explicitly the number p of **nonlinear** equality constraints and the number q of **nonlinear** inequality constraints. The total number of constraints will be computed automatically based on the provided values of p, q and the number of rows of the A, C matrices when linear constraints are present as well.

III) Test the optimization routine on an academic example.
To test the routine, you can use the following optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} \sin\left((\boldsymbol{x}-\boldsymbol{s})/3\right)^T \sin\left((\boldsymbol{x}-\boldsymbol{s})/3\right)$$
$$\text{s.t.}$$
$$A\boldsymbol{x} = \boldsymbol{b}$$
$$e^{-x_1} + a_{\text{eq}} - x_2 = 0$$
$$C\boldsymbol{x} \geq \boldsymbol{d}$$
$$-((x_1 - a_{\text{ineq}})^2 + (x(2,1) - a_{\text{ineq}})^2) + b_{\text{ineq}}^2 \geq 0$$

where a_{eq}, a_{ineq}, $b_{\text{ineq}} \in \mathbb{R}^+$ and $A, \boldsymbol{b}, C, \boldsymbol{d}$ are parameters to be chosen. This problems features a non-convex cost and both linear and nonlinear constrains. Try to change cost and constraints and the solver initialization and parameters, and visualize what happens. Since the example is in two dimensions, you can also plot the level curves of the cost function. An example of results is shown in Figures 1-2. See also the attached, commented code.

IV) Use the developed routine to solve a Finite Horizon Optimal Control Problem for the nonlinear vehicle model of laboratory session A.
The problem is to maximize the distance covered while satisfying the limits of a double-turn track. Try to formulate the FHOCP and solve it with the SQP code. You will have to choose how to model the track, the total time, the sampling time, input and state constraints. As a suggestion, include in the cost function not only the maximum distance, but also penalty terms on input variations and terminal conditions in order to obtain a sensible trajectory. An example of final outcome is reported

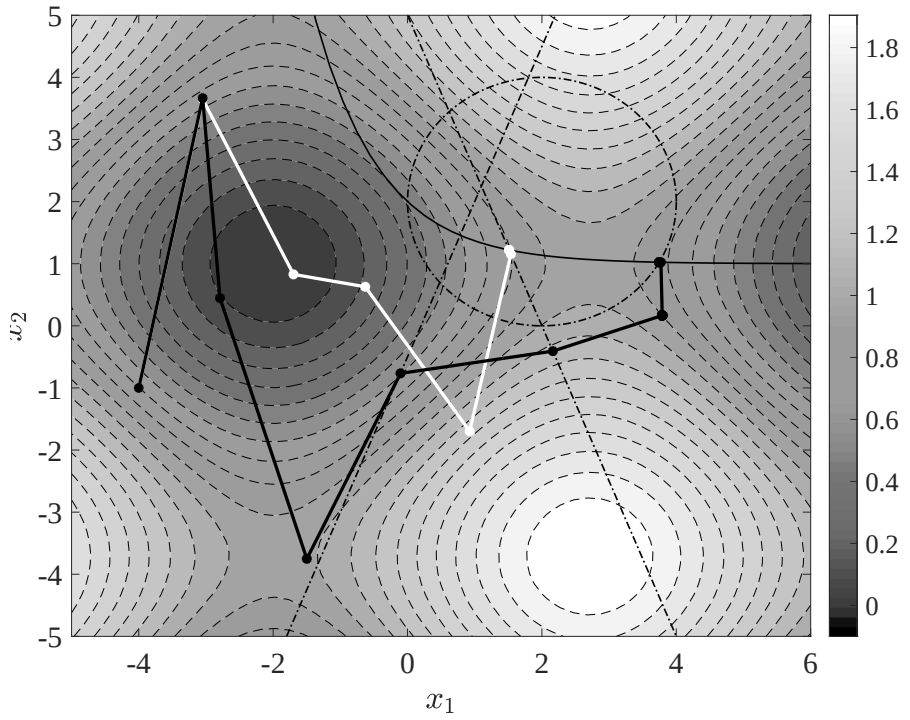

Figure 1: Academic test example with a nonlinear equality constraint (thin, solid black line), and both linear and nonlinear inequality constraints (dash-dotted lines). Thick white line: constrained Gauss-Newton iterations. Thick black line: BFGS iterations.

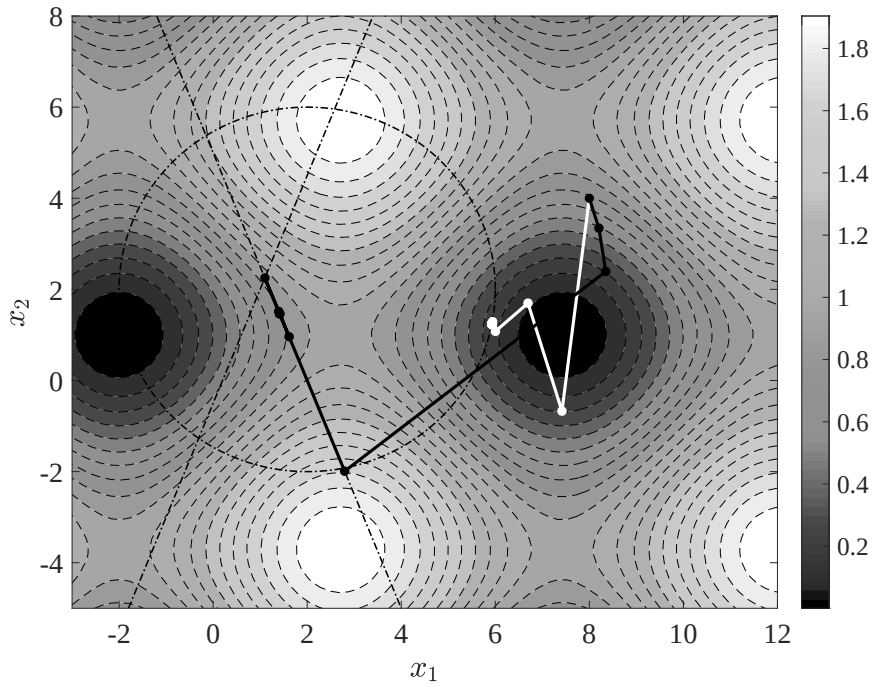

Figure 2: Academic test example with both linear and nonlinear inequality constraints (dash-dotted lines). Thick white line: constrained Gauss-Newton iterations. Thick black line: BFGS iterations.

in Figure 3. For a reference, see the attached, commented code.

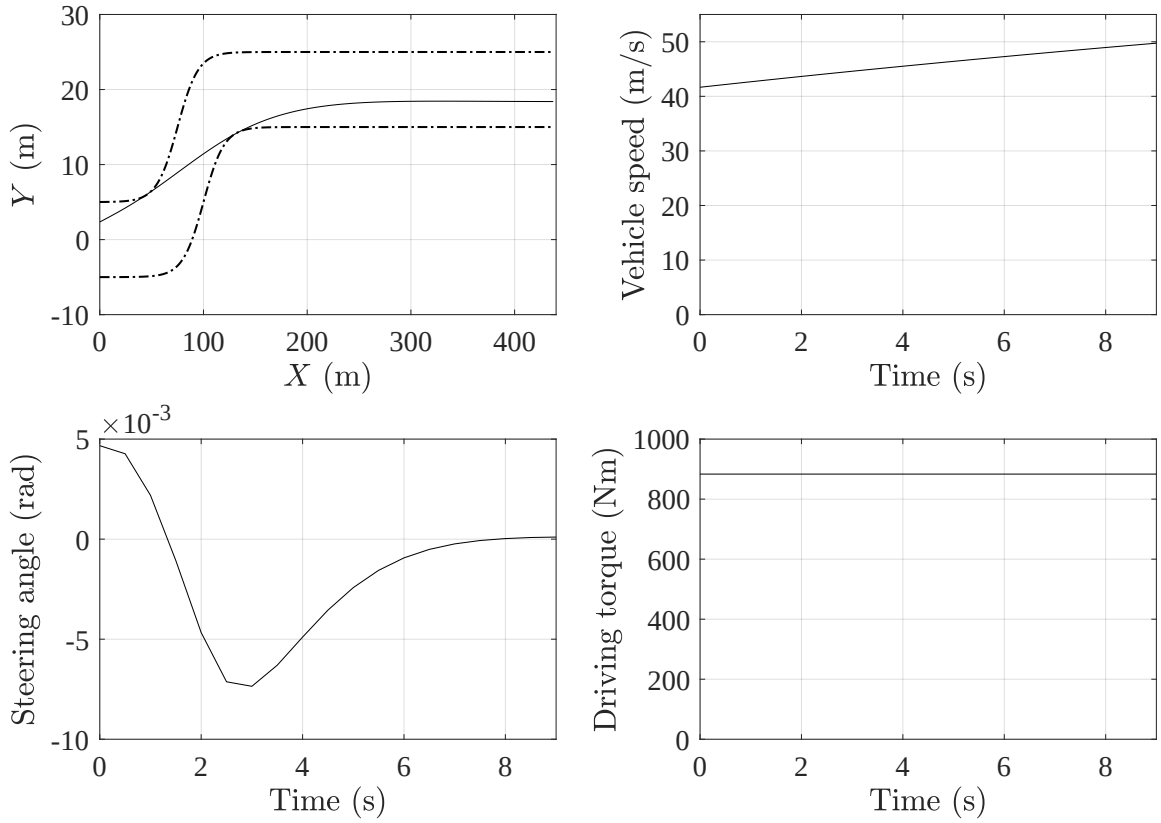

Figure 3: Solution of a FHOCP maximizing the traveled distance over a double-turn track for the road vehicle of Laboratory session A, obtained with SQP.