An Application of Sequential Convex Programming to Time Optimal Trajectory Planning for a Car Motion

Quoc Tran Dinh* and Moritz Diehl*

Abstract— **This paper proposes an iterative method for solving non-convex optimization problems which we call sequential convex programming (SCP) and an application of our method to time trajectory planning problem for a car motion. Firstly, we introduce a formulation of the motion of a car along a reference trajectory as an optimal control problem. We use a special convexity based formulation path employing coordinates and a change of variables technique. Secondly, we use a direct transcription to transform this optimal control problem into a large scale optimization problem which is "nearly" convex. Then, SCP method is applied to solve the resulting problem. Finally, numerical results and comparison with sequential quadratic programming (SQP) and interior point (IP) methods are reported.**

1. Introduction

Time optimal control problems with geometric path appear frequently in mechanical engineering and industrial applications of robotic manipulators [1], [3], [6], [7], [9], [12]. The popular solution approach is to transform the optimal control problems using a direct method into a finite dimensional optimization problem that is in general non-convex and large dimensional. Then optimization algorithms such as sequential quadratic programming or interior point methods are applied to solve the resulting optimization problem. However, if the original problem has some specific structures, e.g.,"near linear" (in terms of small second derivatives), convex objective function, or cone constraints, these methods are too general which can not make use of the specific problem structure directly.

This paper presents a short description of the sequential convex programming method introduced in [8]. Then we apply this method to solve the time optimal trajectory planning problem for a car motion. Our method can be extended to the time optimal trajectory planning for robots control problem [7], [12] with freedom to choose the geometric path. Based on the reference path which we assume to be known in advance and path coordinate system, the dynamic system of the car motion is expressed as a differential-algebraic equation (DAE) with pseudo time, linear differential and nonlinear algebraic parts. We apply a change of variables as in [3], [7], [12], the objective function is transformed to a convex function. The obtained problem is a nonlinear optimal control problem in the path coordinate system.

To solve this optimal control problem, a direct transcription method is applied to transform it into a large scale nonlinear optimization problem. Fortunately, this optimization problem preserves the convexity of the objective function and the "near linearity" of the constraints. If SQP methods or IP methods are applied directly, they do not take into account the structure of this problem. Therefore, we apply SCP method which exploits the specific structure of the problem and then uses freely available software [4], [5], [10], [11] for solving convex subproblems. The numerical results show that our method is more efficient and leads to the solution faster than SQP or IP method in this specific case.

The paper is organized as follows. In Section 1, we describe one way to formulate the time optimal trajectory planning problem for a car motion as an optimal control problem. A brief description of the sequential convex programming method is presented in Section 3. In Section 4, we apply a direct method in optimal control to transform the original problem into a finite dimensional optimization problem. This section also describes condensing techniques to reduce the size of the optimization problem. Preliminary numerical results are reported in the last section to illustrate the efficiency of our method.

2. Problem Formulation

We consider the motion of a car close to a reference trajectory shown in Fig. 1. The car moves along the road based on a reference trajectory from A to B. We suppose that the width of the road is $2b_{max}$, allowing deviation to both sides of the reference trajectory. To formulate the motion

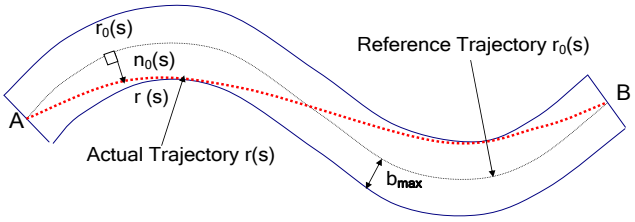

Fig. 1. The motion of a car along a given path.

of the car, we consider the path $r(s)$ with two components $x(s)$ and $y(s)$, given in a Cartesian coordinate system, as a function of a scalar path coordinate (i.e. the arc length s). We denote by $r_0(s) = (x_0(s), y_0(s))$ the reference trajectory of the road. The actual trajectory of the car is written as follows

$$r(s) = r_0(s) + b(s)n_0(s), \tag{1}$$

where $n_0(s)$ is the normal vector of $r_0(s)$ at s and $b(s)$ is the deviation from the reference trajectory at s.

*Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering, Katholieke Universiteit Leuven, Belgium, {quoc.trandinh, moritz.diehl}@esat.kuleuven.be

The path coordinate $b(s)$ determines the spatial geometry of the path, whereas the trajectory's time dependency follows from the relation $s(t)$. Without loss of generality, it is assumed that the trajectory starts at $t = 0$ and ends at $t = T$ such that $s(0) = 0 \le s(t) \le s(T) = 1$. For the actual trajectory $r(s)$, using the chain rule, the velocities $v(s)$ and the accelerations $a(s)$ of the motion can be written by

$$v(s) = \dot{r}(s) = r'(s)\dot{s}, \quad a(s) = \ddot{r}(s) = r'(s)\ddot{s} + r''(s)\dot{s}^2, \quad (2)$$

where $\dot{s} = \frac{ds}{dt}$, $\ddot{s} = \frac{d^2s}{dt^2}$, $r'(s) = \frac{\partial r(s)}{\partial s}$ and $r''(s) = \frac{\partial^2 r(s)}{\partial s^2}$. Taking the first and the second order derivatives with respect to s in (1) and using similar notations, we get

$$r'(s) = r'_0(s) + b(s)n'_0(s) + b'(s)n_0(s), \quad (3)$$
$$r''(s) = r''_0(s) + b(s)n''_0(s) + 2b'(s)n'_0(s) + b''(s)n_0(s). \quad (4)$$

Substituting (3) and (4) into the last term of (2), it implies that

$$a(s) = [r'_0(s) + b(s)n'_0(s) + b'(s)n_0(s)]\ddot{s} + [r''_0(s) + b(s)n''_0(s) + 2b'(s)n'_0(s) + b''(s)n_0(s)]\dot{s}^2. \quad (5)$$

For convenience of notations, we denote the constant terms by

$$p_0(s) := r'_0(s), \quad p_1(s) := n'_0(s), \quad p_2(s) := n_0(s),$$
$$q_0(s) := r''_0(s), \quad q_1(s) := n''_0(s). \quad (6)$$

Then the acceleration $a(s)$ in (5) is expressed as follows

$$a(s) = [p_0(s) + b(s)p_1(s) + b'(s)p_2(s)]\ddot{s} + [q_0(s) + b(s)q_1(s) + 2b'(s)p_1(s) + b''(s)p_2(s)]\dot{s}^2. \quad (7)$$

In this paper we consider the case that the acceleration $a(s)$ is controlled by a driver in two directions, the forward direction along the path of a car and the normal direction; for turning on the left and on the right. By using $n(s)$ as the normal vector of $r(s)$, and $D(s) := \begin{bmatrix} r'(s)^T/\|r'(s)\| \\ n(s)^T \end{bmatrix}$, the acceleration $a(s)$ is constrained by

$$\underline{a} \le D(s)a(s) \le \bar{a}, \quad (8)$$

where $\underline{a} = (\underline{a}_t, \underline{a}_n)$ and $\bar{a} = (\bar{a}_t, \bar{a}_n)$ are the lower and the upper bounds of the acceleration a with respect to the two directions.

The time optimal trajectory planning problem minimizes time T of the car motion from A to B. Using the same technique as in [7], [12] we can write:

$$T = \int_0^T dt = \int_{s(0)}^{s(T)} \frac{ds}{\dot{s}}. \quad (9)$$

If we denote $e(s) := \ddot{s}$ and $f(s) := \dot{s}^2$ then, by the chain rule, we have

$$\dot{f}(s) = f'(s)\dot{s} = 2\dot{s}\ddot{s} = 2e(s)\dot{s}. \quad (10)$$

By assumption $\dot{s} > 0$ almost everywhere, it follows from (10) that

$$f'(s) = 2e(s), \quad (11)$$

Introducing new variables $\tilde{a}(s) := D(s)a(s)$, $c(s) := b'(s)$ and $d(s) := c'(s)$ and substituting them into (7) and (9), we obtain the following optimal control problem:

$$\min_{a(\cdot),b(\cdot),c(\cdot),d(\cdot),e(\cdot),f(\cdot)} \int_0^1 \frac{ds}{\sqrt{f(s)}} \quad (12)$$
$$\text{s.t. } \tilde{a}(s) = D(s)[p_0(s) + b(s)p_1(s) + c(s)p_2(s)]e(s) \quad (13)$$
$$+ [q_0(s) + b(s)q_1(s) + 2c(s)p_1(s) + d(s)p_2(s)]f(s)$$
$$b'(s) = c(s),\ c'(s) = d(s),\ f'(s) = 2e(s), \quad (14)$$
$$b(0) = b_0,\ b(1) = b_T, \quad (15)$$
$$f(0) = \dot{s}_0^2,\ f(1) = \dot{s}_T^2, \quad (16)$$
$$f(s) \ge 0, \quad (17)$$
$$-b_{\max} \le b(s) \le b_{\max}, \quad (18)$$
$$\underline{a} \le \tilde{a}(s) \le \bar{a}, \quad (19)$$

for all $s \in [0,1]$. The notations b_0 and b_T present the starting and finishing positions of the car, respectively. In most cases, $\dot{s}_0$ and $\dot{s}_T$ can be taken by zero.

Problem (12)-(19) can be regarded as an optimal control problem in differential algebraic form (DAE), with pseudo time s, three differential state variables b, c and f, one two-dimensional algebraic state variable $\tilde{a}$ and two input variables e and d.

We say that an optimal control problem is convex if its objective function is convex, the dynamic system is linear and the other constraints are convex. The following lemma shows that if $b_{\max} = 0$ then Problem (12)-(19) is convex [12].

Lemma 2.1: If $b_{\max} = 0$ then Problem (12)-(19) is convex.

Proof: Note that the dynamic system (14) and the constraints (15)-(19) of Problem (12)-(19) are linear. If $b_{\max} = 0$ then it follows from (18) that $b(s) = 0$ for all $s \in [0,1]$. Using (14) we have $c(s) = 0$ which implies $d(s) = 0$ for $s \in [0,1]$. From definition of $D(s)$, it is easy to show that $D(s) = \begin{bmatrix} r'_0(s)/\|r'_0(s)\| \\ n_0(s) \end{bmatrix}$ which is independent from $e(\cdot)$ and $f(\cdot)$. Substituting $b(s) = 0$, $c(s) = 0$, $d(s) = 0$ and $D(s)$ into (13) we obtain

$$\tilde{a}(s) = D(s)[p_0(s)e(s) + q_0(s)f(s). \quad (20)$$

which is linear. The lemma is proved. ■

Note that for fixed b, c and d, the remaining problem is always convex.

3. Sequential Convex Programming Algorithm

This section provides a short description of the line search sequential convex programming algorithm. Let us start by considering the following non-convex optimization problem:

$$\begin{cases} \min\limits_{w \in \mathbf{R}^{n_w}} & F(w) \\ \text{s.t.} & G(w) = 0,\ w \in \Omega, \end{cases} \quad (21)$$

where $F : \mathbb{R}^{n_w} \to \mathbb{R}$ is convex, $\Omega \subset \mathbb{R}^{n_w}$ is nonempty closed convex and $G : \mathbb{R}^{n_w} \to \mathbb{R}^m$ is nonlinear and smooth. An explicit representation of Ω is

$$\Omega := \left\{ w \in \mathbb{R}^{n_w} \mid H_j(w) \le_K 0,\ j = 1, \ldots, p \right\}, \quad (22)$$

where K is a pointed, closed convex cone and H_j are generalized convex functions. In our problem the convex set Ω consists of the linear constraints and, as we will show later, of second order cone constraints.

SCP algorithm is an iterative and local optimization method which starts from a given initial point w^0 and generates a sequence $\{w^p\}$ as follows

$$w^{p+1} = w^p + \alpha_p \Delta w^p, \tag{23}$$

where $\alpha_p > 0$ is a step size and Δw^p is a search direction. SCP methods compute the search direction Δw^p by solving a convex subproblem which obtains by linearizing the nonlinear equality constraints $G(w) = 0$ and preserves the convexity of the objective function and the remaining constraints. The convex subproblem which we need to solve at the iteration p becomes

$$\begin{cases} \min\limits_{\Delta w} & F(w^p + \Delta w) \\ \text{s.t.} & J_G(w^p)\Delta w + G(w^p) = 0,\ w^p + \Delta w \in \Omega, \end{cases} \tag{24}$$

where $J_G(w^p) := (\frac{\partial G(w^p)_i}{\partial w_j})_{ij}$ is the Jacobian of G at w^p. To compute the step size α_p we apply a line search procedure based on a merit function, e.g., l_1-penalty function. The SCP method is briefly described as follows

Algorithm A (*SCP algorithm*):

Initialization: Take an initial point $w^0 \in \Omega$ and set $p := 0$.

Iterations: For $p = 0, 1, 2, \ldots$, execute the following steps:

Step 1: Solve the convex subproblem (24) to get a search direction Δw^p.

Step 2: If $\|\Delta w^p\| \le \varepsilon$ with some tolerance $\varepsilon > 0$ then stop. Otherwise, use a backtracking line-search procedure based on l_1-penalty function $\phi_1(\cdot;\mu_p)$ to get a step length α_p satisfying the Armijo condition (26) (see below).

Step 3: Update $w^{p+1} := w^p + \alpha_p \Delta w^p$ and penalty parameter μ_p using (27). Increase p by 1 and go back to Step 1.

Recall the l_1-penalty function $\phi_1(w;\mu)$ defined by

$$\phi_1(w;\mu) := F(w) + \mu \|G(w)\|_1, \tag{25}$$

where $\mu > 0$ is a penalty parameter. Using ϕ_1 as a merit function, we compute the step size $\alpha_p \in (0,1]$ from the Armijo condition based on backtracking procedure

$$\phi_1(w^p + \alpha_p \Delta w^p; \mu) \le \phi_1(w^p;\mu) + c_1 \alpha_p D(\phi_1(w^p;\mu); \Delta w^p), \tag{26}$$

where $c_1 \in (0,1)$ is a given constant and $D(\phi_1(w^p;\mu);\Delta w^p)$ is the directional derivative of $\phi_1(\cdot;\mu)$ along the search direction Δw^p. To update the penalty parameter μ^p we using the following formula:

$$\mu_p \ge \mu_{lb} := \frac{J_F(w^p)^T \Delta w^p}{(1-\rho)(\|G(w^p)\|_1)}, \tag{27}$$

for some $\rho \in (0,1)$. Since Ω is convex and $\alpha_p \in (0,1]$, the sequence $\{w^p\}_{p\ge 0}$ generated by Algorithm A always belongs to Ω.

SCP method can be considered as a generalization of a constrained Gauss-Newton method for solving least squares problems. The objective function in the SCP method is convex which is more general than quadratic form. Note that the local rate of convergence of SCP method is linear [8], but if the contraction factor of linear convergence is sufficiently small then SCP algorithm converges sufficiently fast in practice.

4. Numerical Solution

A. Direct transcription of optimal control problem

This subsection introduces a short description of a direct transcription method applied to the optimal control problem (12)-(19). It approximates the infinite dimensional problem by a large scale finite dimensional optimization problem.

In order to transform the optimal control problem (12)-(19) into a finite dimensional optimization problem, we first discretize the path coordinate s on $[0,1]$ by $0 = s_0 < s_1 < \cdots < s_N = 1$, with $N+1$ grid points s_k. Then, we parameterize the continuous function $d(\cdot)$ as a piecewise constant function $\hat{d}(\cdot)$ where

$$\hat{d}(s) = d^k := d(s^k),\ \forall s \in [s_k, s_{k+1}),\ 0 \le k \le N-1. \tag{28}$$

From the differential equation (14), by integrating $\hat{d}$, it follows from (28) that $c(\cdot)$ can be approximated by a piecewise linear function $\hat{c}(\cdot)$. Similarly, the function $b(\cdot)$ can be approximated by a piecewise quadratic functions $\hat{b}(\cdot)$ by integrating $\hat{c}(\cdot)$. We also approximate $e(\cdot)$ by a piecewise constant function $\hat{c}(\cdot)$ and $f(\cdot)$ by $\hat{f}(s) = \int \hat{c}(s)ds$, a piecewise linear function.

The function $\tilde{a}(\cdot)$ is evaluated at the middle points $s_{k+1/2} = (s_k + s_{k+1})/2$ of $[s_k, s_{k+1}]$ for all $k = 0, \ldots, N-1$ which means that $\tilde{a}^k := \tilde{a}(s_{k+1/2})$. All the coefficient functions $p_0(\cdot)$, $p_1(\cdot)$, $p_2(\cdot)$, $q_0(\cdot)$ and $q_1(\cdot)$ are evaluated at the middle points $s_{k+1/2}$ with their values denote by p_0^k, p_1^k, p_2^k, q_0^k and q_1^k for all $k = 0, \ldots, N-1$, respectively. From the relationship between b, c and d, we can compute directly $c^{k+1/2} := \hat{c}(s_{k+1/2}) = (c^k + c^{k+1})/2$ and $b^{k+1/2} := \hat{b}(s_{k+1/2}) = b^k + c^k \Delta s_k/2 + d^k \Delta s_k^2/8$. Similarly, from $f'(s) = 2e(s)$, we have $f^{k+1/2} := \hat{f}(s_{k+1/2}) = (f^k + f^{k+1})/2$.

The direct transcription method applied to the optimal control problem (46)-(19) results in the following steps:

First, we approximate the objective function $J(\cdot)$ by

$$\begin{aligned} J(\tilde{a}, b, c, d, e, f) &:= \int_0^1 \frac{ds}{\sqrt{f(s)}} = \sum_{k=0}^{N-1} \int_{s_k}^{s_{k+1}} \frac{ds}{\sqrt{f(s)}} \\ &\approx \sum_{k=0}^{N-1} \int_{s_k}^{s_{k+1}} \frac{ds}{\sqrt{\hat{f}(s)}} = \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{f^k} + \sqrt{f^{k+1}}} \end{aligned} \tag{29}$$

where $\Delta s_k = s_{k+1} - s_k$ for all $k = 0, \ldots, N-1$.

Second, we discretize the dynamic system of (14) by integrating the piecewise constant functions $\hat{d}(\cdot)$ and $\hat{e}(\cdot)$ to get

$$\begin{cases} b^{k+1} - b^k = \Delta s_k c^k + \frac{1}{2}\Delta s_k^2 d^k \\ c^{k+1} - c^k = \Delta s_k d^k \\ f^{k+1} - f^k = 2\Delta s_k e^k, \end{cases} \tag{30}$$

for all $k = 0, \ldots, N-1$.

Finally, the equality and inequality constraints are discretized at $s_{k+1/2}$ and matrix $D(b,c)$ is also discretized with respect to $b^{k+1/2}, c^{k+1/2}$ by $D^k := D(b^{k+1/2}, c^{k+1/2})$. In summary, the resulting optimization problem is written as follows

$$\min_{\tilde{a},b,c,d,e,f} \quad \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{f^k} + \sqrt{f^{k+1}}} \tag{31}$$

$$\text{s.t.} \quad \tilde{a}^k = D^k[p_0^k + p_1^k b^{k+1/2} + p_2^k c^{k+1/2}]e^k + D^k[q_0^k + q_1^k b^{k+1/2} + 2p_1^k c^{k+1/2} + p_2^k d^k]f^{k+1/2}, \tag{32}$$

$$b^{k+1} - b^k = \Delta s_k c^k + \frac{1}{2}\Delta s_k^2 d^k, \tag{33}$$

$$c^{k+1} - c^k = \Delta s_k d^k, \tag{34}$$

$$f^{k+1} - f^k = 2\Delta s_k e^k, \tag{35}$$

$$f^k \geq 0, \tag{36}$$

$$f^0 = \dot{s}_0^2,\ f^N = \dot{s}_T^2, \tag{37}$$

$$b^0 = b_0,\ b^N = b_T, \tag{38}$$

$$-b_{\max} \leq b^k \leq b_{\max}, \tag{39}$$

$$a_{\min}^k \leq \tilde{a}^k \leq a_{\max}^k, \tag{40}$$

for all $k = 0, \ldots, N-1$.

For simplicity, let us define

$$z := (\tilde{a}^0, \tilde{a}_2^0, \ldots, \tilde{a}_1^{N-1}, \tilde{a}_2^{N-1}, \ldots, f^0, \ldots, f^N)^T. \tag{41}$$

as a new variable in $\mathbb{R}^{7(N+2)}$. Introducing new functions

$$F(z) := \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{f^k} + \sqrt{f^{k+1}}}, \tag{42}$$

and $G(z) := (G^0(z)^T, G^1(z)^T, \ldots, G^{N-1}(z)^T)^T$ where

$$G^k(z) := D^k[p_0^k + p_1^k b^{k+1/2} + p_2^k c^{k+1/2}]e^k + D^k[q_0^k + q_1^k b^{k+1/2} + 2p_1^k c^{k+1/2} + p_2^k d^k]f^{k+1/2} - \tilde{a}^k \tag{43}$$

and a polyhedral set

$$\Omega := \{z \mid z \text{ satisfies constraints (33)-(40)}\}. \tag{44}$$

Then, problem (31)-(40) can be rewritten in the short form

$$\begin{cases} \min\limits_{z \in \mathbb{R}^{7(N+2)}} & F(z) \\ \text{s.t.} & G(z) = 0,\ z \in \Omega. \end{cases} \tag{45}$$

Note that the objective function $F(z)$ of the problem (31)-(40) is convex, the equality constraint $G(z) = 0$ is nonlinear and the other constraints are linear. According to Lemma 2.1, $G(z)$ is linear if b, c and d is fixed, then G is slightly near-linear if b, c and d are small, which can happen if $b_{\max}$ is small. In this case problem (31)-(40) can be solved efficiently by applying SCP algorithm.

B. Compute predefined information for optimization problem

In practice, the coefficients p_0, p_1, p_2, q_0 and q_1 can be computed explicitly in advance. Suppose that the coordinate of $r_0(s)$ in the Cartesian coordinate system is

$$r_0(s) = (x_0(s), y_0(s)). \tag{46}$$

Then, the normal vector $n_0(s)$ is determined as follows

$$n_0(s) = \Big(\frac{y_0'(s)}{\sqrt{x_0'(s)^2 + y_0'(s)^2}}, \frac{-x_0'(s)}{\sqrt{x_0'(s)^2 + y_0'(s)^2}}\Big). \tag{47}$$

If we define a mapping $\theta : [0,1] \to [0, 2\pi]$ such that

$$\begin{cases} \cos\theta(s) = \frac{x_0'(s)}{\sqrt{x_0'(s)^2 + y_0'(s)^2}} \\ \sin\theta(s) = \frac{y_0'(s)}{\sqrt{x_0'(s)^2 + y_0'(s)^2}}, \end{cases} \tag{48}$$

then $n_0(s) = (\sin\theta(s), -\cos\theta(s))$ and $\theta(s) = \arctan(y'(s)/x'(s))$. Using the chain rule, it follows from (6), (46)-(40) that

$$\begin{aligned} p_0(s) &= (x_0'(s), y_0'(s)) \\ p_1(s) &= (\cos\theta(s), \sin\theta(s))\theta'(s), \\ p_2(s) &= (\sin\theta(s), -\cos\theta(s)), \\ q_0(s) &= (x_0''(s), y_0''(s)), \\ q_1(s) &= (-\sin\theta(s), \cos\theta(s))\theta'(s)^2 + (\cos\theta(s), \sin\theta(s))\theta''(s), \end{aligned} \tag{49}$$

where

$$\theta'(s) = \frac{x_0'(s)y_0''(s) - x_0''(s)y_0'(s)}{x_0'(s)^2 + y_0'(s)^2}$$

$$\theta''(s) = \frac{x_0'y_0''' - x_0'''y_0'}{x_0'(s)^2 + y_0'(s)^2} - \frac{2(x_0'y_0'' - x_0''y_0')(x_0'x_0'' + y_0'y_0'')}{(x_0'(s)^2 + y_0'(s)^2)^2}.$$

C. Condensing

Since the convexity is preserved under any linear transformation, to reduce the size of optimization problem (31)-(40) we can condense the linear part of the constraints by eliminating dependent variables. The variables b^k, c^k and f^k can be eliminated recursively from $k = 0$ to $k = N-1$.

We first eliminate variables b^k, c^k and f^k based on the linearity of their constraints. Introducing new variables $\tilde{a} := (\tilde{a}_1^0, \tilde{a}_2^0, \ldots, \tilde{a}_1^{N-1}, \tilde{a}_2^{N-1})^T$, $u := (c^0, d^0, \ldots, d^{N-1})^T$ and $e := (e^0, \ldots, e^{N-1})^T$. Then, using the notation $\bar{G}^k$ for the condensing form of the nonlinear constraints (43), after a lengthy derivation, we obtain the reduced optimization problem as follows:

$$\min_{\tilde{a},u,e} J(e) := \sum_{k=0}^{N-1} \frac{2\Delta s_k}{\sqrt{P_k^T e + \dot{s}_0^2} + \sqrt{P_{k+1}^T e + \dot{s}_0^2}} \tag{50}$$

$$\text{s.t.} \begin{cases} \bar{G}^k(\tilde{a}^k, u, e) = 0,\ k = 0, \ldots, N-1, \\ r^T u = b_T - b_0, \\ q^T e = \dot{s}_T^2 - \dot{s}_0^2, \\ Pe + \dot{s}_0^2 \mathbf{1}_f \geq 0, \\ (-b_{\max} - b_0)\mathbf{1}_b \leq Nu \leq (b_{\max} - b_0)\mathbf{1}_b, \\ \underline{a} \leq \tilde{a}^k \leq \bar{a},\ k = 0, \ldots, N-1, \end{cases} \tag{51}$$

where $J(e)$ is convex, vectors r and q are given, $\mathbf{1}_b$ and $\mathbf{1}_f$ are two vectors whose component is 1 and P, N are two constant matrices.

If we define $w := (a^T, u^T, e^T)^T \in \mathbb{R}^{4N+1}$, $\bar{G}(w) := (\bar{G}^0(\tilde{a}^0, u, e)^T, \ldots, \bar{G}^{N-1}(\tilde{a}^0, u, e)^T)^T$, $F(w) := J(e)$ and other

linear constraints of (50) by Ω then the problem (50)-(51) can be rewritten as

$$\begin{cases} \min_w & F(w) \\ \text{s.t.} & G(w)=0,\ w\in\Omega, \end{cases} \tag{52}$$

which collapses again to the same form as (23).

Note that the sparse structure of the optimization problem (31)-(40) is lost after applying the condensing technique. It is not a disadvantage in this specific case, because we use Sedumi and SDPT3 solvers, which benefits from the reduced problem dimension. Alternatively, we could exploit the sparse structure of problem (31)-(40) as well.

D. Solving the convex subproblem by second order cone programming

To apply the SCP method, Algorithm A, the nonlinear equality constraint $G(w)=0$ is linearized at the current point w^p by

$$J_G(w^p)\Delta w + G(w^p) = 0, \tag{53}$$

where $J_G(w^p)$ is the Jacobian of G at w^p. The Jacobian of $G(w)$ can be computed by

$$J_G(w) := (J_{G^0}(\tilde{a}^0,u,e),\ldots,J_{G^{N-1}}(\tilde{a}^{N-1},u,e)^T)^T, \tag{54}$$

where $J_{G^k}^T = (\frac{\partial \hat{G}^k}{\partial \tilde{a}^k}, \frac{\partial G^k}{\partial u}, \frac{\partial G^k}{\partial e})$ is given explicitly.

To exploit the freely available software for solving second order convex programming (SOCP) such as Sedumi [10], SDPT3 [11], YALMIP [5] or CVX [4], we need to transform the convex subproblem (24) to a SOCP problem (see, [2], [12]).

Introducing new slack variables $v := (v_0,\ldots,v_N)^T$ and $t := (t_0,\ldots,t_{N-1})^T$, the objective function (50) can be represented as a linear objective function

$$J(e,v,t) := 2\sum_{k=0}^{N-1} \Delta s_k t_k, \tag{55}$$

and $2N$ additional second order convex cone constraints

$$\left\| \begin{matrix} 2v_k \\ P_k^T e + \dot{s}_0^2 - 1 \end{matrix} \right\|_2 \le P_k^T e + \dot{s}_0^2 + 1, \tag{56}$$

$$\left\| \begin{matrix} 2 \\ v_k + v_{k+1} - t_k \end{matrix} \right\|_2 \le v_k + v_{k+1} + t_k, \tag{57}$$

for all $k=0,\ldots,N-1$. The convex subproblem (24) in Algorithm A collapses to an SOCP problem which can be solved efficiently by Sedumi or SPDT3 solvers.

5. Numerical Results.

This section presents numerical results which are implemented on a Matlab version 7.7.0 (R2008.b) and running on a PC Desktop Pentium IV (2.6GHz, 512Mb RAM). We perform Algorithm A with different reference trajectory formulations of $r_0(s)$. To illustrate the performance, we present the following cases.

Case 1: The first formula of $r_0(s)$ is taken from [7] (with a scaling factor) as follows:

$$\begin{cases} x_0(s) = 50(s-0.5) \\ y_0(s) = 200s^3 - 300s^2 + 100s. \end{cases} \tag{58}$$

We choose the number of grid points $N=50$ and the parameters $\underline{a}=(-50,-10)^T$, $\bar{a}=(10,10)^T$, $b_{\max}=1$, $\dot{s}_0 = \dot{s}_T = 0$ and $b_0 = b_T = 0$. The sample time is $\Delta s_k = \Delta = (s_T - s_0)/N = 1/N$ for all $k=0,\ldots,N-1$.

Fig. 2 presents the actual trajectory of the car. The car is controlled to move from A to B as fast as possible. This figure shows that the trajectory touches the border of the road at several points. The acceleration $a(\cdot)$ of the

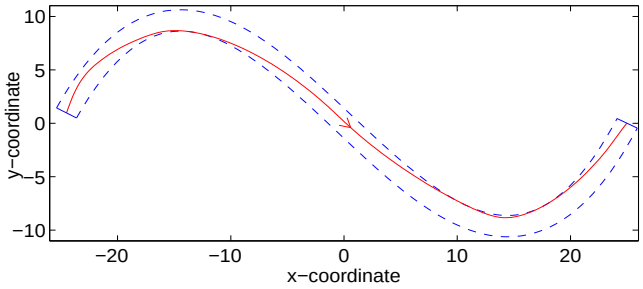

Fig. 2. Actual trajectory of the car motion (Case 1).

car motion is shown in Fig. 3, where the horizontal axis represents the s-coordinate and the vertical axis represents the acceleration in two directions. The dash path indicates the forward acceleration a_t and the dash-dot one is the normal acceleration a_n. Fig. 4 shows the velocity $v(\cdot)$ of the car motion in Case 1.

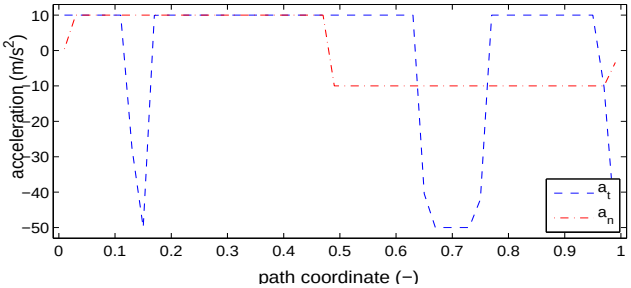

Fig. 3. Acceleration of the car motion (Case 1).

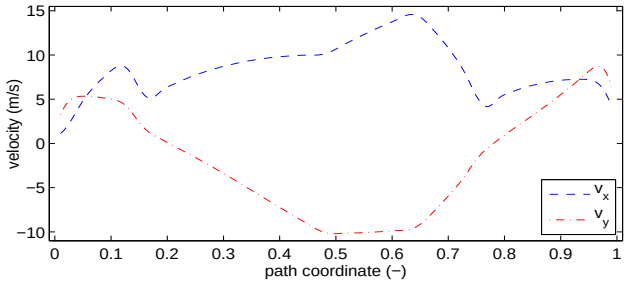

Fig. 4. Velocity of the car motion (Case 1).

Case 2: If we choose

$$\begin{cases} x_0(s) = [20+10\cos(4\pi s)]\cos(2\pi s) \\ x_0(s) = [20+10\cos(4\pi s)]\sin(2\pi s) \end{cases}$$

then the trajectory of the car is plotted in Fig. 5. Fig. 6 and Fig. 7 show the acceleration $a(\cdot)$ and the velocity $v(\cdot)$, respectively.

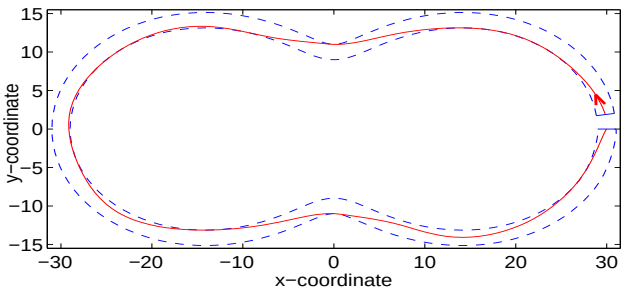

Fig. 5. Actual trajectory based on a circle path (Case 2).

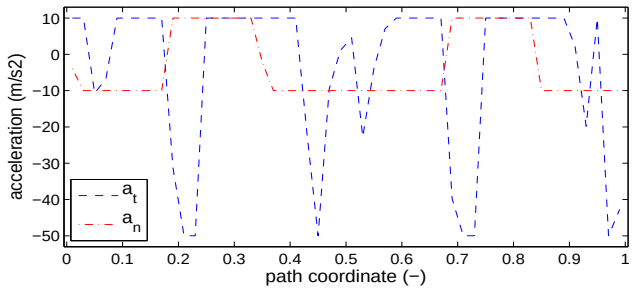

Fig. 6. Acceleration of the car motion (Case 2).

To compare the implementation with SQP and IP methods, we use the standard Matlab's solver FMINCON to solve the resulting optimization problem with the same parameters and the tolerance `Tol`=10^{-3}. The results are reported in Table 5 which present the convergence behavior of SCP, SQP and IP methods for Case 1 with $N = 50$. The notations `Num. of Iter.`, `FirstOrdOpt.` and `ConstrViol.` stand for the number of iterations, first order optimality and constraint violation values, respectively. For Case 2, SQP and IP methods do not converge with the same parameters in Algorithm A.

We performed these three methods with many different reference trajectory formulations and different numbers of grid points N from 20 to 100. The results show that SCP method works better than the others when N is small ($N \leq 50$) and takes few iterations to converge to solution. In contrast, SQP and IP methods are unstable (in the sense that they converge in some cases and do not converges in other cases).

6. Acknowledgments

Moritz Diehl is an associate professor at Katholieke Universiteit Leuven. Research supported by Research Council KUL: CoE EF/05/006 Optimization in Engineering(OPTEC), IOF-SCORES4CHEM, GOA/10/009 (MaNet), GOA/10/11, several PhD/postdoc and fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects G.0452.04, G.0499.04, G.0211.05, G.0226.06, G.0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, G.0588.09,G.0377.09, research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, Belgian Federal Science Policy Office: IUAP P6/04; EU: ERNSI; FP7-HDMPC, FP7-EMBOCON, Contract Research: AMINAL. Other: Helmholtz-viCERP, COMET-ACCM.

TABLE I

Convergence behavior of the three algorithms (Case 1).

`Methods`	SCP	SQP	IP
`Num. of Iter.`	10	200	200
`CPU Time`	30.26	71.36	61.61
`FirstOrdOpt.`	0.00212	1.92032	0.00795
`ConstrViol.`	6.4473e-08	57.7236e-4	2.4101e-04

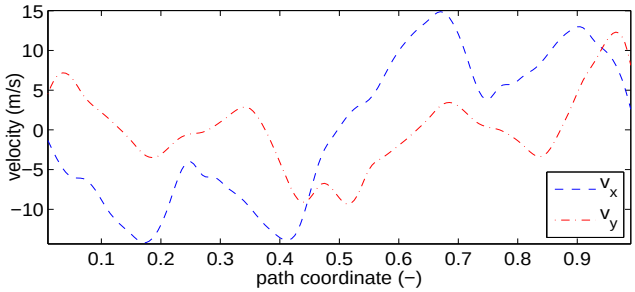

Fig. 7. Velocity of the car motion (Case 2).

References

[1] Aldrich, J.B. and Skelton, R.E., Time-energy optimal control of hyper-actuated mechanical systems with geometric path constraints, *Decision and Control, 2005 and 2005 European Control Conference*, Volume , Issue , 12-15 Dec. 2005, 8246 - 8253.

[2] Boyd S. and Vandenberghe L., *Convex Optimization*, Cambridge University Press (2004).

[3] H. Choset, W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementation*, MIT Press, June 2005.

[4] Grant M., Boyd S. and Ye Y., *CVX user's guide for cvx version 1.2*, http://www.stanford.edu/ boyd/cvx/.

[5] Löfberg J, *YALMIP : A toolbox for modeling and optimization in MATLAB*, IEEE Int. Sym. on Computer Aided Control Systems Design (Taiwan) 284-289 (2004).

[6] Ma Shugen and Watanabe Mitsuru, Time-optimal control of kinematically redundant manipulators with limit heat characteristics of actuators, *Advanced Robotics*, Volume 16, Number 8, 2002 , pp. 735-749(15)

[7] F. Pfeifer and R. Johanni, A concept for manipulator trajectory planning, IEEE Journal of Robotics and Automation, vol. RA-3, no. 2, p. 115-123 (1987).

[8] Quoc T. D. and Diehl M. , *A Line Search Sequential Convex Programming Algorithm for Solving Non-convex Optimization Problems*, in preparation, (2009).

[9] Z. Shiller and H.-H. Lu, Computation of path constrained time optimal motions with dynamic singularities, *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 114, pp. 34-40, March 1992.

[10] Sturm J. F., *Using Sedumi 1.02, a Matlab Toolbox for Optimization over symmetric cones*, http://sedumi.ie.lehigh.edu/ (2001).

[11] Tütünkü R. H., Toh K. C. and Todd M. J., *SDTP3 - a Matlab software package for semidefinite-quadratic-linear programming version 3.0* (2001).

[12] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter and M. Diehl, *Time-optimal path tracking for robots: a convex optimization approach*, IEEE Transactions on Automatic Control, conditionally accepted (2008).