A Class of Non-Linear Time Scaling Functions For Smooth Time Optimal Control Along Specified Paths

Arun Kumar Singh[2] and K.Madhava Krishna [1]

Abstract— **Computing time optimal motions along specified paths forms an integral part of the solution methodology for many motion planning problems. Conventionally, this optimal control problem is solved considering piece-wise constant parametrization for the control input which leads to convexity and sparsity in the optimization structure. However, it also results in discontinuous control trajectory which is difficult to track. Thus, in this paper we revisit this time optimal control problem with the primary motivation of ensuring a high degree of smoothness in the resulting motion profile. In particular, we solve it with continuity constraints in control and higher order motion derivatives like jerk, snap etc. It is clear that such constraints would necessitate the use of time varying control inputs over the commonly used piece-wise constant form.**

The primary contribution of the current work lies in the introduction of a C^∞ class of time scaling functions represented as parametric exponentials. This in turn allows us to represent time varying control inputs as products of parametric exponential and a polynomial functions. We present the motivation behind adopting such representation of time scaling function over more common polynomial forms, both from mathematical as well as implementation standpoint. We also show that the proposed representation of time scaling function and control input leads to a very simple optimization structure where most of the constraints are linear. The non-linearity has a quasi-convex structure which can be reformulated into a simple *difference of convex* form. Thus, the resulting optimization can be efficiently solved through *sequential convex programming* where, at each iteration, the constraints in *difference of convex* form are further simplified to more conservative linear constraints.

I. INTRODUCTION

To reduce the complexity of a general motion planning problem, it is often broken down into two hierarchical steps. At the first step, smooth kinematically feasible, collision free paths are computed, and then at the second step, motion profiles along it are computed subject to velocity, acceleration bounds and higher level constraints such as dynamic collision avoidance. Some such works can be found in, [1], [2]. Thus, computing time optimal motions along a given path subject to some generic set of constraints is critical to various motion planning problems.

Time optimal control along specified paths decomposes a general optimal control problem into just two variables, velocity and acceleration along the given path. These two variables can respectively be considered as the pseudo state and control input. Many existing works like [3], [4], [5], [6] solves this optimal control problem considering piece-wise constant form for the control input (path acceleration). As shown in [5], [6], such representation leads to convexity and sparsity in the structure of the resulting optimization. Convex optimizations are not only computationally fast, but also various high fidelity open source solvers like [7] exist for rapid prototyping. But this computational gain provided by piece-wise constant form comes at the cost of discontinuous control trajectory which are difficult to track [8]. Although [9] uses piece-wise constant control, it includes additional jerk constraints to smoothen the transition of control inputs between the maximum and minimum bound.

In this paper, we revisit this problem of time optimal control along specified paths with the primary motivation of ensuring high degree of smoothness in the resulting motion profiles. This is accomplished by enforcing continuity in the control inputs, as well as higher order motion derivatives like jerk, snap, etc. We show that the proposed framework is capable of ensuring motion continuity upto any desired order, depending on the smoothness of the given path. It is clear that control and motion continuity constraints would necessitate the use of time varying control input and consequently non-linear time scaling function. Most commonly, non-linear time scaling functions are represented as parametric polynomial functions like B-Spline [10], [11], [12]. However, we show in section III that, for the problem of path constrained time optimal control, there is a necessity to depart from this polynomial representation. To give a quick preview here, we show that it is difficult to enforce the necessary positive definite condition on time scaling functions which are based on polynomial representations.

The primary contribution of the current work lies in the introduction of a globally C^∞ class of non-linear time scaling functions, represented as parametric exponentials. It comes as a generalisation of a globally C^0 class of functions, introduced in our earlier works [2], [13] for differentially constrained motion planning problems and used in [14] within the context of path constrained time optimal framework. The parametric exponential based time scaling function allows us to represent time varying control inputs as product of a parametric exponential and a polynomial function. We show that such representation leads to a very simple optimization structure where most of the constraints are linear. The non-linearity has a quasi-convex structure which can be reformulated into a simple *difference of convex* (**DC**) form [16]. Although theoretically any non-convex constraint can be represented in **DC** form, obtaining the exact mathematical representation is not trivial. Moreover the complexity of the optimization is dictated by the chosen **DC** representation [17]. We show that the **DC** reformulation

[1] Robotics Research Center, IIIT-Hyderabad India, [2] Bio-Medical Robotics Lab, BGU, Israel arunkima@post.bgu.ac.il, mkrishna@iiit.ac.in The research was partially supported by the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Initiative of Ben-Gurion University of Negev, Israel

presented in the current work has the simplest form. The resulting optimization is solved through *sequential convex programming* where, at each iteration the **DC** constraints are further simplified to a more conservative linear constraints. We also compare our proposed time scaling function with the current state of the art in non-linear time scaling function [12] and show significant improvement in terms of optimality and fidelity of the resulting optimization.

The rest of the paper is organized as follows: Section II describes the problem formulation along the lines of [5] and [6] in the form of an optimization problem. Section III introduces the proposed C^∞ class of scaling functions and subsequently presents a simplification of the optimization problem . Section IV explains the solution procedure through *sequential convex programming*. Section V presents some comparative results and discussions.

II. Problem Formulation

Without loss of generalization, we consider a path in Euclidean space i.e $\mathbf{X}(u) = (x(u), y(u), z(u))^T$, where $u \in [u_0 \;\; u_f]$ is the path variable. The objective is to compute time optimal motion profiles along the path $\mathbf{X}(u)$ subject to actuator constraints. Like [4], [5] we assume that the actuator constraints can be transformed to equivalent bounds on acceleration. This assumption holds for many common robotic systems like manipulators [4] and non-holonomic wheeled mobile robots [14]. Moreover even for robotic systems like quadrotors time optimal planning can be done with acceleration bounds as actuator constraints [18]. Thus, building on the philosophy of [3] and [4] to compute time optimal motions, at each point along the path, we seek to bring the velocity profile of the robot as close as possible to the maximum velocity limit subject to acceleration bounds. The formulation presented in this section has been done assuming a free flying robot in 3D space like a UAV. Extension to manipulator, non-holonomic systems etc. can be easily done.

The total velocity $v(t)$ and acceleration $a(t)$ of a robot operating in 3D space can be written as

$$v(t) = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2}, a(t) = \sqrt{\ddot{x}(t)^2 + \ddot{y}(t)^2 + \ddot{z}(t)^2} \tag{1}$$

$\dot{x}(t), \ddot{x}(t)$ and similarly others represents individual velocity and acceleration components respectively. They can be derived from path derivatives $x^{'}(u), x^{''}(u)$ by transforming the path variable from u to the time variable t which results in the following expressions.

$$\dot{\mathbf{X}}(t) = \mathbf{X}^{'}(u)\frac{du}{dt}, \ddot{\mathbf{X}}(t) = (\frac{du}{dt})^2\mathbf{X}^{''}(u) + \mathbf{X}^{'}(u)\frac{d^2u}{dt^2} \tag{2}$$

The above expression suggests that the velocity components are derived by scaling the path derivatives with the function $\frac{du}{dt}$ which is called the scaling function and decides the transformation from the path variable u to the time variable t. $\frac{du}{dt}$ and $\frac{d^2u}{dt^2}$ is equivalent to the concept of path velocity and acceleration in [3], [4], [6] . In all these works, path acceleration, which can be considered as a pseudo control input is piece wise constant. In [5], a piece-wise constant form of $\frac{d^2u}{dt^2}$ is used. But as shown in the next section, in the current proposed work $\frac{du}{dt}$ and $\frac{d^2u}{dt^2}$ belongs to globally C^∞ class of functions.

Using the notation $\dot{u} = \frac{du}{dt}$ and substituting (2) in (1) we get

$$v(t(u)) = \dot{u}\sqrt{v(u)}, \;\; v(u) = (x^{'}(u))^2 + (y^{'}(u))^2 + (z^{'}(u))^2 \tag{3}$$

As stated above, time optimal control along a specified path reduces to that of bringing the velocity profile as close as possible to maximum velocity limit. In other words, the following objective function is to be minimized.

$$J = \int_{u_0}^{u_f} (\dot{u}\sqrt{v(u)} - v_{max}(u))^2 \tag{4}$$

The objective function,(4) has to be minimized with respect to the following constraints

$$\dot{u}\sqrt{v(u)} \leq v_{max}(u) \tag{5}$$

$$\dot{u} \in \mathcal{C}^\infty \tag{6}$$

$$\begin{cases} \dot{u}(u_0) = \dot{s}_0, \dot{u}(u_f) = \dot{s}_f \\ \ddot{u}(u_0) = \ddot{s}_0, \ddot{u}(u_f) = \ddot{s}_f \\ \dddot{u}(u_0) = \dddot{s}_0, \dddot{u}(u_f) = \dddot{s}_f \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ u^{(m)}(u_0) = s_0^{(m)}, u^{(m)} = s_f^{(m)} \end{cases} \tag{7}$$

$$|x^{''}(u)\dot{u}^2 + x^{'}(u)\ddot{u}| \leq \ddot{x}_{max}(u) \tag{8}$$

$$|y^{''}(u)\dot{u}^2 + y^{'}(u)\ddot{u}| \leq \ddot{y}_{max}(u) \tag{9}$$

$$|z^{''}(u)\dot{u}^2 + z^{'}(u)\ddot{u}| \leq \ddot{z}_{max}(u) \tag{10}$$

The constraint (5) ensures that the velocity profile is always less than the limit curve. The constraint (6) ensures that the resulting time optimal motion profile can be made continuous upto any desired order depending on the smoothness of the given path $\mathbf{X}(u)$. The constraint (7) ensures boundary constraints on the scaling function and its derivatives. It is clear that appropriate boundary values for the scaling function $\dot{u}$ and its derivatives can be chosen to enforce boundary constraints on corresponding trajectory derivatives. For example, boundary values for $\dot{u}$ and $\ddot{u}$ can be chosen appropriately to satisfy the boundary value constraints on $\dot{\mathbf{X}}(t)$ and $\ddot{\mathbf{X}}(t)$. Inequalities (8)-(10) represent bound constraints on the individual acceleration components.

The path derivatives like $x^{'}(u), x^{''}(u)$ and others are determined by the given path $\mathbf{X}(u)$ and hence, the only variable of the optimization problem (4)-(10) is the pseudo control input function $\ddot{u}$ (since $\dot{u}$ is itself determined by $\ddot{u}$). Although at the moment the objective function and constraints do not reveal any special structure as such, but the reformulation presented in the next section where $\dot{u}$ is represented as C^∞ class of parametric exponential functions, allows us to convert the objective function and constraints into quadratic and linear form respectively.

III. Reformulation through C^∞ class of Parametric Exponential Scaling Functions

We propose the following parametric exponential based time scaling function belonging to C^∞ class of functions

$$\dot{u} = e^{P(u)} \Rightarrow \ddot{u} = \dot{u}^2 P^{'}(u) \tag{11}$$

Where, $P(u)$ is a l^{th} order polynomial in terms of variable u and can be represented in many convenient forms, two

of which are represented below. In (12), u^k is the regular polynomial basis while $B_k^l(u)$ is the basis in Bernstein polynomial form. a_k is the coefficient associated with the basis.

$$P(u) = \sum_{k=0}^{l} a_k u^k, \quad P(u) = \sum_{k=0}^{l} B_k^l(u) a_k \tag{12}$$

As mentioned earlier, many works like [10], [11], [12] adopt a polynomial representation for both $\dot{u}$ and $\ddot{u}$. As can be seen from (11), our representation is unique and quite different from the existing polynomial representations like B-Spline. To understand the necessity for this departure, recall that the scaling function $\dot{u}$ decides the transformation between the path variable u and the time variable t. Since time cannot reverse itself, this transformation between the path and time variable needs to be monotonically increasing function. Consequently, $\dot{u} > 0$ should be maintained throughout the interval of the path variable variable u. This in turn means that only positive definite functions can be a valid choice for $\dot{u}$. Now, it is challenging to construct positive definite polynomials. In fact as shown in [15], ensuring positive definiteness of a polynomial amounts to solving matrix inequalities, which although convex are not easy to solve. Thus, works like [10], [12] adopt a more simple but approximate approach where, $\dot{u} > 0$ is imposed as an additional constraint at sufficient number of grid points along the path. This results in a set of simple linear constraints. Although computationally useful, this approach is highly problem specific and cannot be relied upon. In particular, the number of grid points sufficient to ensure $\dot{u} > 0$ would depend on the given path, the velocity and acceleration bounds along it and cannot be known a priori. We also highlight this critical shortcoming in section V, where we show loss of positive definiteness across various problem instances.

Based on the above discussion, it is clear that the parametric exponential function proposed in (11) is an apt choice for representing $\dot{u}$ since it satisfies the positive definiteness constraint by construction. Although one can construct many positive definite functions, the proposed choice is motivated by the fact that it leads to a very simple optimization structure with constraints in linear and **DC** form. This is an improvement over polynomial constraints obtained in optimization proposed in [10], [11]. Further, we also show in section V, that the performance of the minimum time optimization formulated in section II, is significantly better with the proposed parametric exponentials (11) as compared to polynomial functions proposed in a recent work [12]. Finally, at this juncture it is also worth reiterating that irrespective of the number of parameters, the proposed time scaling function is capable of ensuring arbitrary degree of smoothness; a characteristic that cannot be matched by their polynomial counterparts.

With the necessity for the parametric exponential based time scaling function firmly established, we now present a reformulation of the objective function and constraints (4)-(10) using (11) and (12). However, before proceeding it is worth pointing out that the time scaling function (11) satisfies constraint (6) by construction.

A. Reformulating Velocity Constraints

To reformulate the velocity level constraint, we first discretize the path interval $[u_0 \;\; u_f]$ into $n+2$ grid points as $u_0, u_1, u_2, u_3 .... u_n, u_f$. Next, evaluating the velocity level constraint (5) at the grid points, substituting expression of $\dot{u}$ and taking logarithmic transformation, results in following n inequalities

$$\begin{aligned} C_{vi} = \log(\dot{u}(u_i)) - \log(s(u_i)) \leq 0 \\ \Rightarrow C_{vi} = P(u_i) - \log(s(u_i)) \leq 0, \forall i = 1, 2, 3 .... n \end{aligned} \tag{13}$$

$$s(u) = \frac{v_{max}(u)}{\sqrt{(x^{'}(u))^2 + (y^{'}(u))^2 + (z^{'}(u))^2}} \tag{14}$$

The first term in n velocity level inequality constraints C_{vi} is a logarithm of an exponential function $\dot{u}$. Hence it is linear in terms of variables a_k. The second term is constant and thus the velocity constraints represents linear inequalities in terms of polynomial parameters a_k.

B. Reformulation of Boundary value Constraints

Reformulation of boundary value constraints (7) depends on the following lemma

Lemma 3.1: $u^{(m)}(u_0) = s_0^{(m)}, u^{(m)}(u_f) = s_f^{(m)}$ is equivalent to $P^{(m-1)}(u_0) = P_0^{(m-1)}$, $P^{(m-1)}(u_f) = P_f^{(m-1)}$ i.e., the boundary value of the scaling function and its derivatives can be enforced by choosing appropriate boundary values for the polynomial $P(u)$ and its derivatives.

Proof: Lets start with $\dot{u}$. Using its definition from (11), we get

$$\begin{aligned} \dot{u}(u_o) = \dot{s}_0, \dot{u}(u_f) = \dot{s}_f \Rightarrow \log(\dot{u}(u_o)) = \log \dot{s}_0, \\ \log(\dot{u}(u_f)) = \log \dot{s}_f \\ \Rightarrow P(u_o) = \log \dot{s}_0 = P_0, P(u_f) = \log \dot{s}_f = P_f \end{aligned} \tag{15}$$

Now considering that $\ddot{u} = \dot{u}^2 P^{'}(u)$, we get the following equalities

$$\begin{aligned} \ddot{u}(u_0) = \ddot{s}_0, \ddot{u}(u_f) = \ddot{s}_f \\ \Rightarrow P^{'}(u_0) = \frac{\ddot{s}_0}{\dot{s}_0^2} = P_0^{'}, P^{'}(u_f) = \frac{\ddot{s}_f}{\dot{s}_f^2} = P_f^{'} \end{aligned} \tag{16}$$

Similarly $\dddot{u} = \dot{u}^3((2P^{'}(u)^2 + P^{''}(u))$ and thus we obtain the following equalities

$$\begin{aligned} \dddot{u}(u_0) = \dddot{s}_0, \dddot{u}(u_f) = \dddot{s}_f \\ \Rightarrow P^{''}(u_0) = \frac{\dddot{s}_0}{\dot{s}_0^3} - (2P_0^{'})^2 = P_0^{''}, \\ P^{''}(u_f) = \frac{\dddot{s}_f}{\dot{s}_f^3} - (2P_f^{'})^2 = P_f^{''} \end{aligned} \tag{17}$$

We can proceed in a similar manner and derive the boundary value $P_0^{(m-1)}$ and $P_f^{(m-1)}$ required to enforce the boundary constraints on $u^{(m)}$

■

In the light of the above lemma, the boundary value constraints on the scaling function and its derivatives can be reformulated in the following manner

$$C_{bound} = \begin{cases} P(u_o) = P_0, P(u_f) = P_f \\ P^{'}(u_0) = P_0^{'}, P^{'}(u_f) = P_f^{'} \\ ..... \\ P^{(m-1)}(u_o) = P_0^{(m-1)}, P^{(m-1)}(u_f) = P_f^{(m-1)} \end{cases} \tag{18}$$

It can be easily noted that the above modified boundary constraints are linear in terms of the parameters a_k.

C. Reformulating Acceleration Constraints

Similar to velocity constraint, evaluating acceleration constraint (8) at the grid points, and taking logarithmic transformation results in the following inequalities

$$\begin{aligned} \log(\dot{u}(u_i)) + 0.5 g_{xi}(a_k) - 0.5\log(\ddot{x}_{max}(u_i)) \leq 0, \quad (19) \\ \Rightarrow P(u_i) + 0.5 g_{xi}(a_k) - 0.5\log(\ddot{x}_{max}(u_i)) \leq 0 \\ \forall i = 1, 2, ..n \end{aligned}$$

$$g_{xi}(a_k) = \log(|x^{''}(u_i) + P^{'}(u_i)x^{'}(u_i)|) \tag{20}$$

Similar expressions can be written for y and z component of the acceleration bound constraints.

The first terms in each acceleration level constraints are same as the velocity level constraints and thus are linear in terms of parameter a_k. The second term in the i^{th} constraint is a logarithmic function in terms of variables a_k. This logarithmic non-linearity has a special structure. To understand this further, consider the surface of the i^{th} constraint shown in figure 1(a). It shows the structure of the function $g_{xi}(.)$ for $P(u) = a_1 u + a_2 u^2$. As can be seen from the figure, it consists of two concave surfaces on either side of the line passing through a continuum of singular points, satisfying the following equation.

$$x^{''}(u_i) + P^{'}(u_i)x^{'}(u_i) = 0 \tag{21}$$

The logarithmic non-linearity in the i^{th} constraint is quasi-convex in nature since any level set of its will be characterized by intersection of linear inequalities. A detailed proof is omitted here because of lack of space. Quasi-convexity forms the intuition behind the following final modification to the x component of the acceleration bound constraints (19).

$$C_{axi} = \begin{cases} g_{xi}(a_i) \leq \log(\mathcal{G}_{xi}) \\ \Rightarrow x^{''}(u_i) + P^{'}(u_i)x^{'}(u_i) - \mathcal{G}_{xi} \leq 0 \\ \Rightarrow -x^{''}(u_i) - P^{'}(u_i)x^{'}(u_i) - \mathcal{G}_{xi} \leq 0 \\ \log(\dot{u}(u_i)) + 0.5\log(\mathcal{G}_{xi}) - 0.5\log(\ddot{x}_{max}(u_i)) \leq 0 \\ \Rightarrow P(u_i) + 0.5\log(\mathcal{G}_{xi}) - 0.5\log(\ddot{x}_{max}(u_i)) \leq 0 \\ \forall i = 1, 2, 3...n \end{cases} \tag{22}$$

In (22) the slack variable $\mathcal{G}_{xi}$ is used to split the i^{th} acceleration constraint into three separate inequalities. Similar expressions can be derived for the y and z component as C_{ayi} and C_{azi} respectively.

It can be easily noted that the first two inequalities are linear in terms of parameter a_k. The third inequality has a linear term followed by a purely concave non-linearity. As stated in [17], a purely concave function f has the simplest *difference of convex* (**DC**) representation in the form $0 - f$. Thus, the third inequality in (22) represents **DC** constraints in its simplest form.

D. Reformulating Minimum Time Optimization Problem

The minimum time optimization problem can now be formulated as the following.

$$\begin{aligned} \min J_{mod} \quad (23) \\ C_{vi} \leq 0, C_{axi} \leq 0, C_{ayi} \leq 0, C_{azi} \leq 0, \forall i = 1, 2..n \\ C_{bound} = 0 \end{aligned}$$

$$J_{mod} = \sum_{i=1}^{i=n} [\log(\dot{u}(u_i)) - \log(s(u_i))]^2 \tag{24}$$

J_{mod} as given in (24) has been obtained from (4) by approximating the integral through summation and by noting that minimizing an objective $(ax - b)^2$ is similar to minimizing $(\log(x) - \log(\frac{b}{a}))^2$. It can be easily noted that the objective function (24) is quadratic in terms of parameters a_k of the polynomial $P(u)$.

We next describe the solution procedure for the optimization (23).

IV. Solving the Optimization Problem

An optimization problem with purely concave non-linearity represents a special case of *difference of convex programming problem* **DCP** [16]. *Sequential convex programming* (**SCP**) [16] routine is an efficient choice for solving such problems since the notion of trust region usually associated with **SCP** is not relevant for **DCP** problems. In **SCP** procedure, at each iteration, the convex part of the problem is preserved while the non-convex part is replaced with a convex approximation around the solution obtained at the previous iteration. The resulting convex problem can be solved very efficiently and quickly. The **SCP** procedure is continued till some stopping criteria is met.

A. Convex Approximation

The non-convexity in optimization problem (23) arises out of purely concave logarithm functions in the acceleration constraints C_{axi}, C_{ayi} and C_{azi}. Let $\mathcal{G}_{xi}^k$, $\mathcal{G}_{yi}^k$ and $\mathcal{G}_{zi}^k$ represent the slack variables at the end of k^{th} iteration. At the $(k+1)^{th}$ iteration, the concave part is replaced with following affine approximation around the solution obtained at the k^{th} iteration.

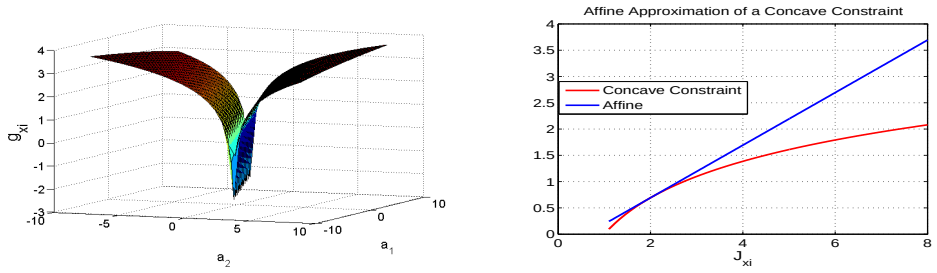
$$\log(\mathcal{G}_{xi}) \simeq \log(\mathcal{G}_{xi}^k) + (\frac{1}{\mathcal{G}_{xi}^k}(\mathcal{G}_{xi} - \mathcal{G}_{xi}^k)) \tag{25}$$

Similar expressions can be written for y and z component as well. The affine approximation is shown in figure 1(b). As it can be seen from the figure, affine approximation to a concave non-linearity acts as a global upper bound. This implies that the feasible region of relaxed problem with approximate affine constraints is contained in that of the original optimization problem.

B. Approximate Quadratic Programme

The affine approximation presented above results in following quadratic programme with only linear constraints at the $(k+1)^{th}$ iteration of **SCP**.

$$\begin{aligned} \min J_{mod} \quad (26) \\ s.t, C_{vi} \leq 0, C_{axi}^{k+1} \leq 0, C_{ayi}^{k+1} \leq 0, C_{azi}^{k+1} \leq 0, \forall i = 1, 2, 3..n \\ C_{bound} = 0 \end{aligned}$$



(a) (b)

Fig. 1. (a): Plot showing the surface of logarithmic non-linearity $g_{xi}(.)$. As it can be seen, the surface consists of two concave parts around a continuum of singular points. It is easy to verify that the non-linearity is actually quasi-convex in nature. (b): Affine approximation of a concave constraints. Affine approximation acts as a global upper bound for the original concave non-linearity. It implies that feasible region of the optimization problem with approximate affine constraints is contained in the feasible region of the optimization with the original non-linear concave constraint.

The **SCP** procedure involving affine approximation and quadratic programming is continued till $|J_{mod}^{k+1} - J_{mod}^{k}| < \lambda$, for some user defined small constant λ. The **SCP** procedure converges within two to three iterations for the optimization problem (23).

C. Examples

Two sample outputs of the minimum time optimization are shown in figures 2(a)-2(d). In accordance with the theory of time optimal motions, at any given instant, either the velocity constraints or the acceleration level constraints should be saturated. This is followed quite closely in 2(a)-2(d). For example, consider figures 2(a)-2(c) where, in the interval $u \in [2\ \ 3.3]$, y component of the acceleration constraint is saturated at the maximum limit. Similarly, in the interval $u \in [3.3\ \ 4.3]$, the scaling function $\dot{u}$ curve lies on the $s(u)$ curve, suggesting the saturation of the velocity level constraints (13). In the interval $u \in [4.5\ \ 4.8]$, x and y component of the acceleration constraints are saturated at the minimum limit, while in the interval $u \in [5.4\ \ 10.0]$, the velocity level constraints are saturated. Similar trend of constraint saturation can be seen in the figures 2(b)-2(d) as well. It can also be observed from the figures that there exist points, where none of the constraints are saturated. This loss in optimality is attributed to discretization issues and can be reduced by increasing the resolution of discretization, although at the cost of increased computation time.

Besides resolution of discretization, the quality of solution of the minimum time optimization (23) depends on the degree of the polynomial $P(u)$. The above discussed results were obtained for a 31^{st} order $P(u)$. Performance of the optimization with various other degree polynomials are summarized in tables I and II. As can be seen, the objective function value decreases with increase in the order of the polynomial $P(u)$. However, the reduction in the objective function is also accompanied by increase in the computation time. The computation time reported in tables I and II were obtained for a MATLAB implementation using the open source tool box CVX [19] on a standard PC with an Intel Core 2 Duo CPU 2.93 Ghz and 4.0 GB of RAM. The optimization was solved with 165 grid points which resulted in around 1650 inequality constraints. The number of variables in the optimization depends on the order of the polynomial $P(u)$ and are summarized in the tables I and II. Significant computation gain can be expected by prototyping the code in C++ or Python. The computation times are low enough to suggest that a real time implementation of the optimization can be obtained by solving it over a shorter horizon.

V. Additional Results and Discussions

A. Effect of Control and Higher Order Motion Continuity

In this section, we highlight the effect of enforcing control (acceleration) as well as higher order motion continuity constraints in the time optimal framework. To this end, we compare the output of the optimization (23) with that proposed in our earlier work [14] which solves the problem of time optimal control along specified paths without acceleration continuity constraints. This particular work was chosen because, it uses a time scaling function very similar to that proposed in (11) and thus, has an identical *difference of convex* form as that obtained for optimization (23). The form of time scaling function used in [14] can be represented in the following manner.

$$\dot{u} = pe^{-qu} \tag{27}$$

It can be easily observed that the above scaling function, belongs to the family of scaling functions proposed in (11). In particular (27) is obtained with $P(u) = -qu$. It can also be easily noted that since (27) has been obtained from a lower order polynomial $P(u)$, it has too few parameters for it to be used in an optimization. Thus, in [14], a time scaling function was constructed by piece-wise combination of (27) (ref. figure 3). As can be observed from figure 3, that $\dot{u}$ used in [14] is continuous but not differentiable and thus, consequently has discontinuity in the acceleration profile.

We are now in a position to compare the output of the optimization (23) with that obtained from [14]. To have a fair comparison, the number of free variables, equality and inequality constraints were kept equal in both the optimizations.

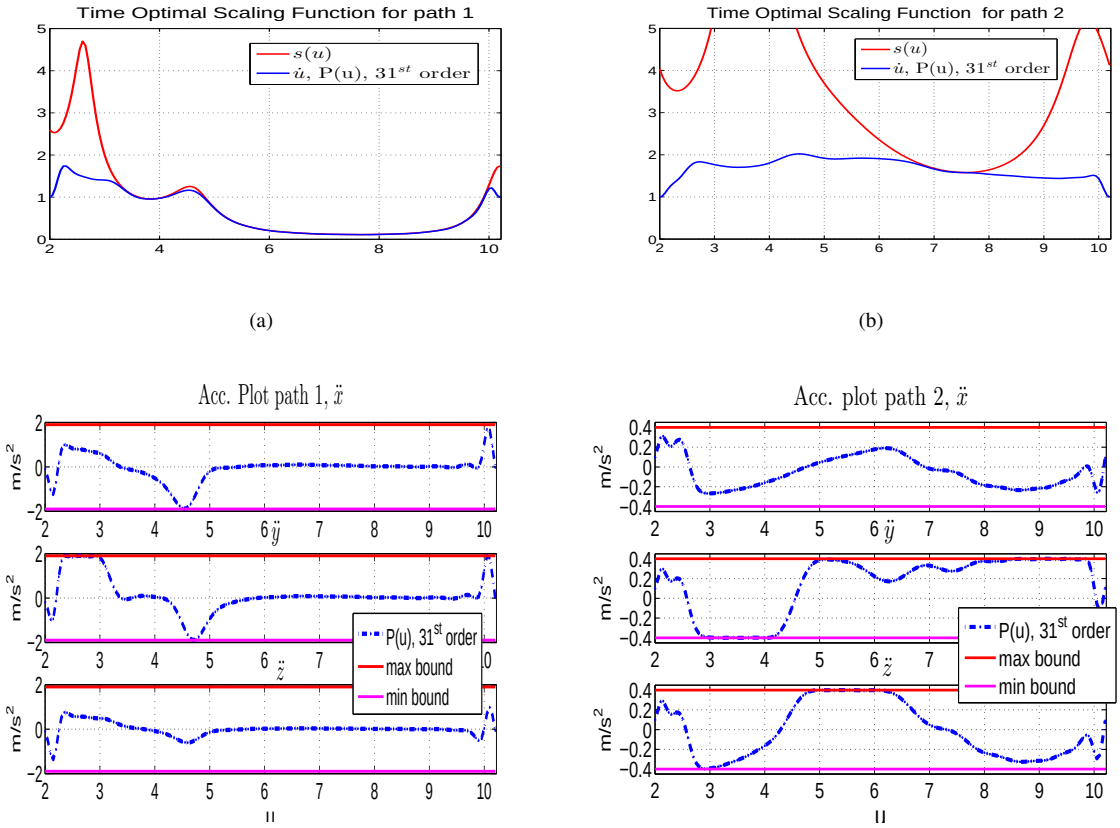


Fig. 2. Sample outputs obtained from solving optimization (23) along two different paths. As can be seen that the resulting scaling function and acceleration profile agrees quite closely with the characteristics of time optimal motion. That is, any one of the constraints is saturated at each instant. As can be seen either the $\dot{u}$ profile is at the limit curve $s(u)$ or the acceleration profile is at the maximum or minimum limit. However, there do exist some points where neither constraints are saturated. This loss in optimality can be attributed to discretization issues and can be reduced by increasing the resolution of discretization.

TABLE I

(23) WITH DIFFERENT ORDERS OF $P(u)$ FOR PATH 1

Order	obj.value	comp.time(ms)	Variables incld. slack
9^{th}	5.19	210	496
15^{th}	4.11	267	502
21^{th}	3.98	400	508
31^{th}	3.7	610	518

TABLE II

(23) WITH DIFFERENT ORDERS OF $P(u)$ FOR PATH 2

Order	obj.value	comp.time(ms)	Variables incld. slack
9^{th}	12.75	230	496
15^{th}	12.06	300	502
21^{th}	11.80	440	508
31^{th}	11.27	680	518

Figures 4(a) and 4(b) compares the output of both the optimizations. A few important things can be easily observed from these two figures. Firstly, the scaling function, $\dot{u}$ curve resulting from optimization (23) is smoother than that obtained through optimization [14]. In particular, $\dot{u}$ resulting from [14] is only continuous in nature as compared to that obtained through optimization (23), which exhibits higher order differentiability. Secondly, $\dot{u}$ profile resulting from both the optimization have similar magnitudes, suggesting similar final objective function values. Thus, the proposed C^{∞} class of scaling functions (11) induces control and motion continuity constraints without significant loss in the optimality of the solution. Thirdly and most importantly the acceleration profile obtained through [14] has abrupt and discontinuous jumps in magnitude. The most prominent discontinuities are highlighted in figure (4(b)). In contrast, the acceleration profile obtained through optimization (23) has a highly smooth evolution.

B. Parametric Exponential Vs Polynomial Time Scaling Functions

In this section, we further expand our comparison between parametric exponential and polynomial based time scaling functions. In particular, the objective of this section is to highlight two important points. Firstly, we show that incorporating $\dot{u} > 0$ as an additional constraint in the minimum time optimization does not reliably ensure positive definiteness of the polynomial based time scaling function. Secondly, we show that for the same number of free vari-

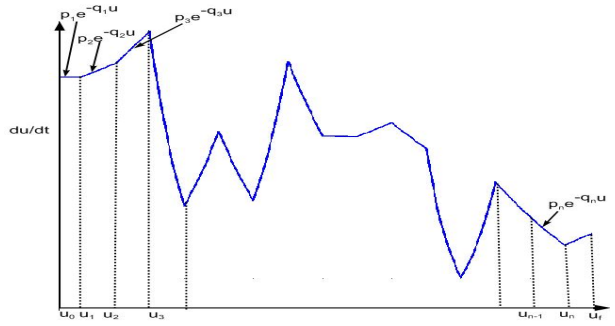
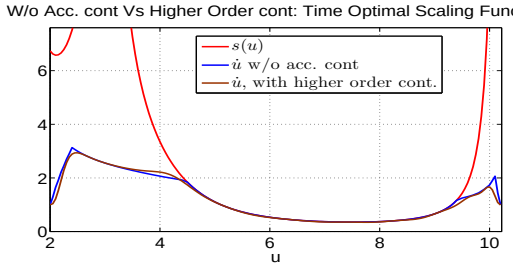
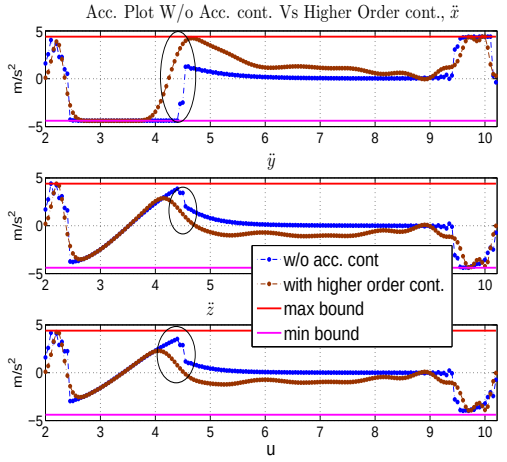


Fig. 3. Time scaling function used in [14]. Discretization of path variable interval into various sub-intervals by the grid points $u_1, u_2, u_3 ... u_n$. An exponential function is fitted in each subinterval. The scaling function $\dot{u}$ is obtained as a continuous combination of the exponential functions. As can be easily observed that $\dot{u}$ is continuous but not differentiable and consequently time optimal motions obtained through it has a discontinuity in the acceleration profile.



(a)



(b)

Fig. 4. (a), (b): Comparison between optimization (23) and that proposed in [14]. Few important things can be easily noted from these figures. Firstly, the $\dot{u}$ curve obtained through solving (23) is much smoother than that obtained through solving [14]. However, both the curves have similar magnitudes which suggests similar final objective function values. Finally, the acceleration profile obtained through solving [14] has abrupt and discontinuous jumps. In sharp contrast, optimization (23) ensures highly smooth evolution of acceleration values.

ables, the performance of the minimum time optimization is significantly better with proposed parametric exponentials as compared to polynomial time scaling functions. To this end, we compare the parametric exponentials proposed in (11) with the following polynomial time scaling function.

$$\dot{u} = \sqrt{P(u)} \Rightarrow \ddot{u} = 0.5P^{'}(u) \tag{28}$$

The above form of scaling function was proposed in the recent work [12] and is significantly better as compared to that used in works like [10], [11] which are also based on polynomial time scaling functions. To be precise, in contrast to polynomial constraints in these cited works, (28) leads to linear constraints. It is straightforward to follow the approach of section III and derive the velocity, acceleration bound constraints, objective function and minimum time optimization problem (23) with the above polynomial based time scaling function and control input.

Figures 5(a)-5(d) compares the output of both the optimizations. As stated before, an additional constraint $\dot{u} > \varepsilon$, where ε is small positive number, has to be incorporated at number of grid points in the minimum time optimization obtained with (28). Please note that ε has to be chosen small enough to make sure it does not conflict with the velocity bound constraints (13). As can be seen from the figures 5(b) and 5(d) that $\dot{u} > \varepsilon$ at a fixed resolution is not sufficient for ensuring positive definiteness of polynomial based time scaling function across various problem instances. To be precise, while a resolution of $\Delta u = 0.005$ was enough to enforce positive definiteness for the example shown in figure 5(b), the same resolution proved insufficient for that shown in figure 5(d). As can be noted from (28) that positive definiteness is lost when $P(u)$ goes to zero or negative, consequently forcing $\dot{u}$ to zero or imaginary values. In figure 5(d), the highlighted points are those where $\dot{u}$ becomes imaginary. For illustrative purposes, the imaginary values are plotted as zeroes in figure 5(d). It is worth mentioning that example shown in (5(d)) had significantly lower velocity bounds that that shown in figure (5(b)). However, the acceleration bounds in both of them were comparable.

Further now, compare figures 5(a)-5(b) and 5(c)-5(d). It can be seen that the time optimal scaling function obtained in parametric exponential form is on an average much closer to $s(u)$ curve than their polynomial counterparts. This in turn means that, for the same number of free variables as decided by the degree of polynomial $P(u)$, parametric exponential (11) leads to lower objective function values than polynomial based time scaling function (28). (refer (24)). However, it is worth mentioning that minimum time optimization with scaling function of the form (28) is generally faster as compared to the optimization obtained with parametric exponentials. This is because of the fact that (28) leads to velocity and acceleration bound constraints in linear form [12]. Also please note that our implementation with (28) is slightly different from [12]. To be precise in contrast to cubic splines in this cited work, our implementation uses higher order polynomials $P(u)$ to ensure continuity in higher order motion derivatives.

VI. Conclusions and Future Work

In this paper, we solved the problem of time optimal control along specified paths with continuity in control and higher order motion derivatives like jerk, snap etc.

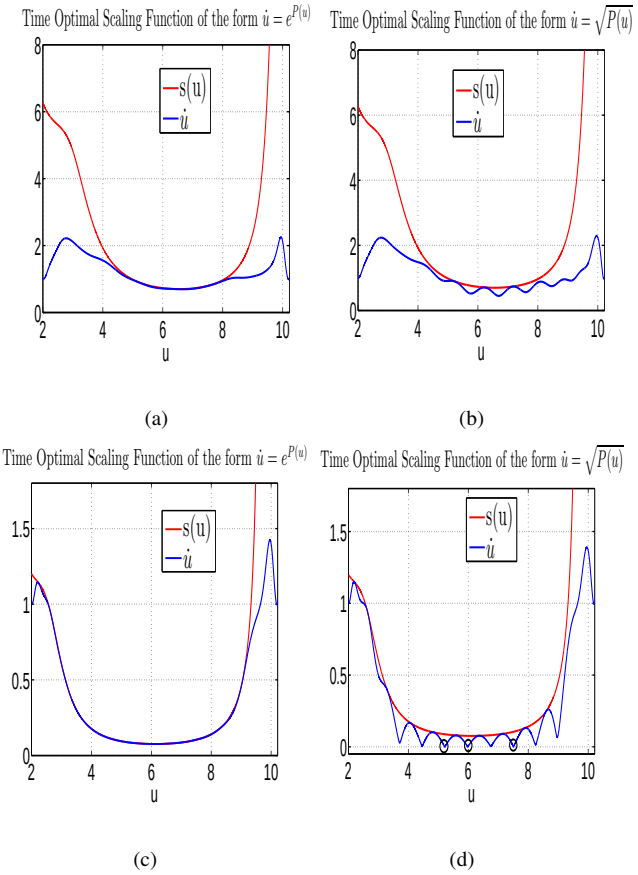


Fig. 5. First Compare figures (b) and (d), which shows that additional constraints in the form $\dot{u} > 0$ does not reliably enforce positive definiteness in the polynomial based time scaling functions. The fundamental issue is that resolution of discretization, i.e, number of grid points at which the constraints $\dot{u} > 0$ needs to be enforced to ensure positive definiteness may vary with problem specifications. As an example, a resolution of 0.005 works for the optimization example shown in (b) but not for the example shown in (d). As can be noted from (28), positive definiteness is lost when $P(u)$ goes to zero or negative and thus forcing $\dot{u}$ to zero or imaginary values. In figure (d), imaginary values of $\dot{u}$ are highlighted as zeroes. Further now compare (a)-(b) and (c)-(d). As can be seen that the time optimal scaling function in the parametric exponential form is much closer to $s(u)$ curve than that based on polynomial functions. This in turn means that for the same number of free variables in the optimization, parametric exponentials leads to lower objective function values than their polynomial counterparts.

We proposed a novel C^∞ class of time scaling functions represented as parametric exponentials. This in turn allowed us to represent time varying control inputs as product of a parametric exponential and a polynomial function. We showed how natural positive definiteness of parametric exponentials makes it a more appropriate choice for a time scaling function as compared to polynomial functions. Moreover performance of minimum time optimization with parametric exponentials was shown to be significantly better than that obtained with polynomial based time scaling functions.

The structure of the minimum time optimization derived with parametric exponential based time scaling function was shown to be very simple with primarily linear constraints. The non-linearity could be easily reformulated into simple *difference of convex* (**DC**) form . A *sequential convex programming* approach was adopted for solving the optimization where, at each iteration, the (**DC**) constraints were further simplified to more conservative linear constraints.

The current work can be extended in various directions. For example, it is easy to incorporate bounds on higher derivatives like jerk. It is not difficult to show that the jerk bound constraints would be quadratic and thus would also have a natural **DC** representation [16]. We are also extending the framework to include force and torque constraints.

REFERENCES

[1] Peng, Jufeng, and Srinivas Akella. "Coordinating multiple robots with kinodynamic constraints along specified paths." The International Journal of Robotics Research 24.4 (2005): 295-310.

[2] Singh, A. K., and Krishna, K. M. (2013, December). Reactive collision avoidance for multiple robots by non linear time scaling. In Proc. of IEEE CDC 2013 (pp. 952-958).

[3] Shiller, Zvi. "On singular time-optimal control along specified paths." Robotics and Automation, IEEE Transactions on 10.4 (1994): 561-566.

[4] Kunz, Tobias, and Mike Stilman. "Time-Optimal Trajectory Generation for Path Following with Bounded Acceleration and Velocity." Proceedings of the 2012 Robotics: Science and Systems Conference. Vol. 8. 2012.

[5] Hauser, Kris. "Fast interpolation and time-optimization on implicit contact submanifolds." Robotics: Science and Systems. 2013.

[6] Verscheure, Diederik, et al. "Time-optimal path tracking for robots: A convex optimization approach." Automatic Control, IEEE Transactions on 54.10 (2009): 2318-2327.

[7] Dahl, Joachin, and L. Vandenberghe. "Cvxopt: A python package for convex optimization." Proc. eur. conf. op. res. 2006.

[8] Guarino Lo Bianco, Corrado, and Oscar Gerelli. "Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints." Robotics, IEEE Transactions on 27.6 (2011): 1144-1152.

[9] Diehl, M., Debrouwere, F., De Schutter, J., Dinh, Q. T., Swevers, J., Pipeleers, G., and Van Loock, W. (2013). Time-Optimal Path Following for Robots With Convex-Concave Constraints Using Sequential Convex Programming. Ieee Transactions On Robotics, 29(EPFL-ARTICLE-196131), 1485-1495.

[10] Kanehiro, F., Suleiman, W., Lamiraux, F., Yoshida, E., and Laumond, J. P. (2008, September). Integrating dynamics into motion planning for humanoid robots. In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on (pp. 660-667). IEEE.

[11] Bouktir, Y., M. Haddad, and T. Chettibi. "Trajectory planning for a quadrotor helicopter." Control and Automation, 2008 16th Mediterranean Conference on. IEEE, 2008.

[12] Zhang, Q., Li, S. R., and Gao, X. S. (2013, June). Practical smooth minimum time trajectory planning for path following robotic manipulators. In American Control Conference (ACC), 2013 (pp. 2778-2783). IEEE.

[13] Singh, A. K., Krishna, K. M., and Saripalli, S. (2012, October). Planning trajectories on uneven terrain using optimization and nonlinear time scaling techniques. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on (pp. 3538-3545). IEEE.

[14] Bharath Gopalakrishnan, Arun Kumar Singh and K Madhava Krishna " Time Scaled Collision Cone based Trajectory Optimization Approach for Reactive Planning in Dynamic Environments" to appear in *IROS 2014*.

[15] Pablo A. Parrilo. Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization. PhD thesis, California Institute of Technology, May 18 2000

[16] Boyd, Stephen. "Sequential convex programming." Lecture Notes, Stanford University (2008).

[17] Thach, P. T., and Konno, H. (1997). On the degree and separability of nonconvexity and applications to optimization problems. Mathematical programming, 77(3), 23-47.

[18] Hoffmann, G. M., Waslander, S. L., and Tomlin, C. J. (2008, August). Quadrotor helicopter trajectory tracking control. In AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii (pp. 1-14).

[19] Grant, M., Boyd, S., and Ye, Y. (2008). CVX: Matlab software for disciplined convex programming.