Proximate Time-Optimal Parameterization of Robot Paths: A Linear-time algorithm

Gerardo Pardo-Castellote * and Robert H. Cannon, Jr. †

Stanford University Aerospace Robotics Laboratory
Stanford, California 94305

Abstract

We present an efficient algorithm to generate proximate time-optimal trajectories for a mechanical manipulator. Given a sequence of via points and constraints in the manipulator's velocity, acceleration and actuator torques, the algorithm produces a continuous trajectory with a continuous acceleration profile. The run-time complexity of the algorithm is linear in the number of via points (or path length). The algorithm has been extensively tested and is currently being used to generate on-line optimal trajectories in a dual-arm robot system. This algorithm is ideally suited to the time-parameterization computer-generated sequences of via points such as the ones produced by many path-planners.

1 Introduction

The problem of obtaining "optimal" trajectories for a robot in the presence of obstacles has been the subject of considerable research. The trajectory, aside from being collision free, must respect the dynamic constraints imposed by the manipulator (e.g. torque, acceleration and/or velocity limits) while being "optimal" with respect to some suitable performance index. Typical performance indices are total travel time, energy consumption or a combination of the two.

Two main approaches have been described in the literature. The first which may be referred as *decoupled* breaks the overall problem in two parts. First a *geometrical* [1] collision free path is obtained using a path planner. This *geometrical* path is described as a continuous function of some parameter "s" (usually path length): $\mathbf{q} = \mathbf{f}(s)$ with, $\mathbf{f} : [0, s_f] \in \mathcal{R} \rightarrow \mathcal{C}$ where $\mathcal{C}$ is the configuration space of the robot. Given the *geometric* path, the next step is to obtain a time parameterization by finding the time history $s(t)$ that minimizes the performance index subject to the dynamic constraints. This latter stage, doesn't change the geometrical path. Hence the name *decoupled*. The algorithms presented in [6, 13, 15, 14, 17] as well as the one presented here follow this approach.

The second, *coupled* approach, attempts to obtain a collision-free path that is also optimal with respect to some performance index without going through an intermediate geometric path. That is, it solves both the path planning and the trajectory-generation problems simultaneously. This is the approach taken in [4, 1, 11, 12]. Its computational complexity (recall that the geometrical path planning problem is itself $\mathcal{NP}$ hard) makes these off-line algorithms.

Sacrificing strict optimality in order to obtain more efficient algorithms has already been suggested in [19, 2], while other authors have focused in improving the performance of true time-optimal methods [18]. In addition to on-line execution, we have addressed the following practical issues:

*Ph.D. Candidate, Department of Electrical Engineering.

†Professor, Dept. of Aeronautics and Astronautics.

[1] The name geometrical is used to stress the fact that time isn't involved.

1. Several instances of the decoupled approach require a continuous geometric path as input. On the other hand many existing planners search a grid in configuration space and so output a sequence of discrete via points. See [7, 5] for details.

2. Many algorithms in the literature produce as output a sequence of desired positions, velocities and accelerations at the sample rate of the controller. In many experiments, there is a need to sample this trajectory at multiple rates-rates that may be unknown to the trajectory-generation algorithm. Moreover, for applications with limited communication bandwidth from trajectory-generator to controller, a continuous trajectory (one that could be described with fewer parameters) is highly desirable.

3. Unless extra constraints are imposed [2], the resulting time-optimal trajectory will be Bang-Bang and thus will have discontinuous accelerations. Many robots exhibit (link or drivetrain) flexibility and discontinuous accelerations would excite the flexible modes in undesirable ways.

4. Many approaches are computationally expensive. Our goal is to do on-line planning and execution of robot tasks and therefore we are willing to trade off strict time-optimality in exchange for a more efficient algorithm. Table 1 summarizes the run-time complexity of several algorithms.

This paper introduces an algorithm that addresses all the above issues. It takes a sequences of via points as an input and, produces a continuous path with continuous accelerations as an output. The algorithm handles manipulator torque, acceleration and velocity constraints. It is extremely efficient having a running time which is $\mathcal{O}(N)$ in the number of via points (and hence in the length of the path).

The remainder of this paper is organized as follows: In section 2 the problem is formally stated and several of the approaches in the literature are reviewed, in section 3 the approximations introduced by the algorithm are presented and discussed. Section 4 introduces the algorithm and discusses its time complexity and section 5 presents the time-parameterization of some example paths. Appendix A describes how we generated the "canonical" paths used for the complexity measurements and for all the examples in this paper.

2 Problem Formulation and Known Results

The decoupled optimal time parameterization (DOTP) problem has been extensively described in the literature [6, 13, 3]. In order to introduce our algorithm and compare it to the existing approaches we need to present summarize some of these well-known facts.

Assuming known robot dynamics and no friction:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{B}(\mathbf{q})[\dot{\mathbf{q}}, \dot{\mathbf{q}}] + \mathbf{C}(\mathbf{q})[\dot{\mathbf{q}}^2] + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{1}$$

Where $\mathbf{q}$ is the vector of generalized coordinates, $\mathbf{M}$ is the mass matrix, $\mathbf{B}$ is the $n \times (n-1)$ matrix of coriolis terms, $\mathbf{C}$ is the $n \times n$ matrix of centripetal terms, $\mathbf{g}(\mathbf{q})$ is the $n \times 1$ matrix of gravity terms and $\boldsymbol{\tau}$ is the torque vector.

A *geometric* path is given as a function $\mathbf{q} = \mathbf{f}(s)$ where the parameter "s" is in some range: $s \in [s_0, s_f]$ has been given as a function of a parameter "s". Typically the function $f(s)$ is required

[2]See for example [15].

to have continuous second derivatives. It is well-known that such a geometric path and the time evolution of the parameter $s = s(t)$ completely determine the full trajectory of the robot and therefore the required torques. That is, given $s = s(t)$ we can use equations (2) and (1) to obtain $\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)$ and $\boldsymbol{\tau}(t)$.

$$\mathbf{q}(t) = \mathbf{q}(s(t)) \;\Rightarrow\; \dot{\mathbf{q}} = \mathbf{f}_s \dot{s} \;\Rightarrow\; \ddot{\mathbf{q}} = \mathbf{f}_s \ddot{s} + \mathbf{f}_{ss} \dot{s}^2 \tag{2}$$

To formulate an instance of the DOTP problem, we also need a performance index and a set of constraints. The performance index $\mathcal{J}$ is a functional $\mathcal{J} = \mathcal{J}[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}]$. Again using (2) and (1) we can write $\mathcal{J} = \mathcal{J}[s, \dot{s}, \ddot{s}]$. For example, the total travel time can be written as: $\mathcal{J} \overset{\text{def}}{=} t_f = \int_{s_0}^{s_f} \frac{ds}{\dot{s}(s)}$. The velocity, acceleration and torque constraints are specified as a set of inequalities $\boldsymbol{\phi}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}) \leq \mathbf{0}$, which can be written as $\boldsymbol{\phi}(s, \dot{s}, \ddot{s}) \leq \mathbf{0}$.

The DOTP problem is the search for the time parameterization $s = s(t)$ with $s(t) \in \mathcal{C}^2[s_0, s_f]$ [3] that minimizes the performance index $\mathcal{J}$ subject to the constraint $\boldsymbol{\phi} \leq \mathbf{0}$. The introduction of the *geometric* path simplifies the problem by replacing the search for vector-valued control history $\boldsymbol{\tau}(t)$ by that of a scalar valued one: $\ddot{s}(t)$.

The approaches in the literature differ in their selection of performance index $\mathcal{J}$, the nature of the constraint $\boldsymbol{\phi} \leq \mathbf{0}$ imposed, and the method used to solve the optimization problem.

reference	perf. index	constraints	method	complexity
Luh el al. [8]	total travel time	constant bounds on $\dot{q}$ and $\ddot{q}$	modified approximate programming	$\mathcal{O}(L)$/iter.
Bobrow el al. [6]	total travel time	$\tau[q, \dot{q}]$ and $\dot{q}[q]$	direct integration	$\mathcal{O}(L^2)$
Shin el al. [16]	total travel time	$\tau[q, \dot{q}]$ and $\dot{q}[q]$	direct integration	$\mathcal{O}(L^2)$
Shin el al. [15]	general	general	dynamic programming	$\mathcal{O}(L)$/iter.
Singh el al. [17]	average of travel time and energy	$\tau[q, \dot{q}]$ and $\dot{q}[q]$	dynamic programming	$\mathcal{O}(L)$/iter.
Pardo et al. this paper	total travel time	$\tau[q, \dot{q}]$, $\dot{q}[q]$ and/or $\ddot{q}[q, \dot{q}]$	direct integration	$\mathcal{O}(L)$

Table 1: **Related approaches to the Decoupled Optimal Time Parameterization Problem.**

Comparison of approaches to the DOTP problem. The notation $\boldsymbol{\tau}[\mathbf{q}, \dot{\mathbf{q}}]$ indicates limits in the torque that may depend on the state $(\mathbf{q}, \dot{\mathbf{q}})$, "general" indicates support for a wide variety of possible performance indices or constraints. In the complexity column L stands for path length. For iterative algorithms that require convergence, we give the complexity per iteration. See sections 4.4 and 4.5 for the complexity column.

In the specific case for which the performance index is the total travel time, and there are only constraints in the velocity, acceleration and torques, it has been shown in [6, 13] that the DOTP problem can be transformed into a simpler one.

[3]The notation $\mathcal{C}^n[a, b]$ denotes the set of functions with continuous nth derivative in the interval [a,b]

2.1 Minimum travel time with limits on actuator torque, acceleration and joint velocities

Using equations (2) and (1) we obtain

$$\begin{aligned} \boldsymbol{\tau}(s,\dot{s}) &= \mathbf{M}(s)(\mathbf{f}_s\ddot{s} + \mathbf{f}_{ss}\dot{s}^2) \\ &\quad + (\mathbf{B}(s)[\mathbf{f}_s;\mathbf{f}_s] + \mathbf{C}(s)[\mathbf{f}_s^2])\dot{s}^2 + \mathbf{g}(s) && (3) \\ \boldsymbol{\tau}(s,\dot{s}) &= \mathbf{m}(s)\ddot{s} + \mathbf{c}(s)\dot{s}^2 + \mathbf{g}(s) && (4) \\ \mathbf{m}(s) &\overset{\text{def}}{=} \mathbf{M}(s)\mathbf{f}_s(s) && (5) \\ \mathbf{c}(s) &\overset{\text{def}}{=} \mathbf{M}(s)\mathbf{f}_{ss}(s) + \mathbf{B}(s)[\mathbf{f}_s;\mathbf{f}_s] + \mathbf{C}(s)[\mathbf{f}_s^2] && (6) \end{aligned}$$

The torque constraints can now be written as,

$$\begin{aligned} \boldsymbol{\tau}_{min}(s,\dot{s}) \le \boldsymbol{\tau}(s,\dot{s}) \le \boldsymbol{\tau}_{min}(s,\dot{s}) && (7) \\ \Longleftrightarrow\ \ddot{s}_{min}(s,\dot{s}) \le \ddot{s} \le \ddot{s}_{min}(s,\dot{s}) && (8) \end{aligned}$$

$$\begin{aligned} \text{Where, } \ddot{s}_{min}(s,\dot{s}) &\overset{\text{def}}{=} \max_{i=1\ldots N_{dof}} \{\delta^i_{min}(s,\dot{s})\} && (9) \\ \ddot{s}_{max}(s,\dot{s}) &\overset{\text{def}}{=} \min_{i=1\ldots N_{dof}} \{\delta^i_{max}(s,\dot{s})\} && (10) \end{aligned}$$

$$\delta^i_{min}(s,\dot{s}) \overset{\text{def}}{=} \begin{cases} \frac{1}{\mathbf{m}^i(s)}\boldsymbol{\tau}^i_{min}(s,\dot{s}) - \mathbf{c}^i(s)\dot{s}^2 - \mathbf{g}^i(s) & \\ \qquad\qquad \text{if } \mathbf{m}^i(s) > 0 & \\ \frac{1}{\mathbf{m}^i(s)}\boldsymbol{\tau}^i_{max}(s,\dot{s}) - \mathbf{c}^i(s)\dot{s}^2 - \mathbf{g}^i(s) & \\ \qquad\qquad \text{if } \mathbf{m}^i(s) < 0 & \\ -\infty \qquad\qquad \text{if } \mathbf{m}^i(s) = 0 & \end{cases} \qquad (11)$$

$$\delta^i_{max}(s,\dot{s}) \overset{\text{def}}{=} \begin{cases} \frac{1}{\mathbf{m}^i(s)}\boldsymbol{\tau}^i_{max}(s,\dot{s}) - \mathbf{c}^i(s)\dot{s}^2 - \mathbf{g}^i(s) & \\ \qquad\qquad \text{if } \mathbf{m}^i(s) > 0 & \\ \frac{1}{\mathbf{m}^i(s)}\boldsymbol{\tau}^i_{min}(s,\dot{s}) - \mathbf{c}^i(s)\dot{s}^2 - \mathbf{g}^i(s) & \\ \qquad\qquad \text{if } \mathbf{m}^i(s) < 0 & \\ \infty \qquad\qquad \text{if } \mathbf{m}^i(s) = 0 & \end{cases} \qquad (12)$$

It is easy to see that using (2), any constraints in the acceleration $\ddot{\mathbf{q}}$, can be expressed in a similar form:

$$\begin{aligned} \ddot{\mathbf{q}}_{min}(s,\dot{s}) \le \ddot{\mathbf{q}}(s) \le \ddot{\mathbf{q}}_{min}(s,\dot{s}) && (13) \\ \Longleftrightarrow\ \ddot{s}_{min}(s,\dot{s}) \le \ddot{s} \le \ddot{s}_{min}(s,\dot{s}) \end{aligned}$$

With the appropriate corresponding definitions for $\ddot{s}_{min}(s,\dot{s})$ and $\ddot{s}_{max}(s,\dot{s})$.

Therefore, constraints in the torques and accelerations can be combined into an inequality constraint of the form $\ddot{s}_{min}(s,\dot{s}) \le \ddot{s} \le \ddot{s}_{min}(s,\dot{s})$. Or using the fact that $\frac{d\dot{s}}{ds} = \frac{\ddot{s}}{\dot{s}}$, they can be written as:

$$\begin{aligned} &\alpha_{min}(s,\dot{s}) \le \tfrac{d\dot{s}}{ds} = \tfrac{\ddot{s}}{\dot{s}} \le \alpha_{max}(s,\dot{s}) \\ &\alpha_{min}(s,\dot{s}) \overset{\text{def}}{=} \ddot{s}_{min}(s,\dot{s})/\dot{s} \text{ For } \dot{s} \neq 0 && (14) \\ &\alpha_{max}(s,\dot{s}) \overset{\text{def}}{=} \ddot{s}_{max}(s,\dot{s})/\dot{s} \text{ For } \dot{s} \neq 0 \end{aligned}$$

Furthermore, constraints in the velocity, if present, can be written as:

$$
\begin{aligned}
&-\dot{\mathbf{q}}_{max}(s) \leq \dot{\mathbf{q}}(s) = \mathbf{f}_s(s)\dot{s} \leq \dot{\mathbf{q}}_{max}(s) \iff \dot{s} \leq \gamma(s) \\
&\gamma(s) \stackrel{\text{def}}{=} \min_{i=1 \dots N_{dof}}\{\frac{\dot{\mathbf{q}}^i_{max}(s)}{|\mathbf{f}^i_s(s)|}\}
\end{aligned}
\tag{15}
$$

In this paper we will make the assumption that the torque limits are such that the manipulator can hold its own weight at any point along the trajectory. That is, $\forall s : \boldsymbol{\tau}(s,0) \leq \mathbf{g}(s) \leq \boldsymbol{\tau}(s,0) \iff \forall s : \alpha_{min}(s,0) \leq 0 \leq \alpha_{max}(s,0)$.

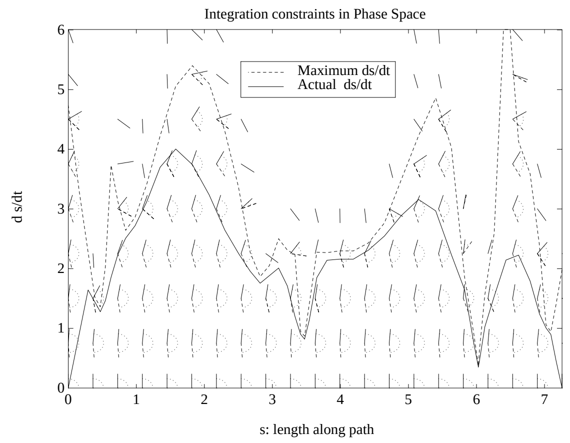

Figure 1: **Integration trajectory in phase space**

This figure shows the proximate-optimal phase-space trajectory. Also shown are the constraints in the phase-space trajectories that correspond to the manipulator dynamic constraints for trajectories along the given geometric path. At each point $(s, \dot{s})$ in phase-space, there is a "wedge" representing the range of allowed slopes of phase-space trajectories trough that point. The curve $\dot{s}_{max}$ separates the region of the phase-space for which there are no allowed phase-space trajectories. The strict limit occurs when the "wedge" "closes". We have imposed a more demanding requirement and disallow regions of phase space for which the wedge doesn't contain the zero slope. Therefore our $\dot{s}_{max}(s)$ curve represents the points for which either the maximum or minimum allowed phase-space slope is zero.

Borrowing the analogy introduced by [16], the constraints of equations (14) can be viewed as a "wedge" associated with each point $(s, \dot{s})$ in phase space. This "wedge" represents the range of allowed slopes of any trajectory that goes through that point in phase space ($\dot{s} = \dot{s}(s)$) and locally satisfies the constraints. Several authors [6, 16] have noted that for each value of s there are values of $\dot{s}$ for which the wedge closes (i.e. $\alpha_{min}(s, \dot{s}) > \alpha_{max}(s, \dot{s})$) meaning there is no phase space trajectory that goes through that point $(s, \dot{s})$ and satisfies the constraints. Figure 1 shows these constraints for an example path. Furthermore, in [16] the authors prove that under fairly general assumptions [4], the "allowed" region for $\dot{s}$ has the form $0 \leq \dot{s}_{max}(s)$. We can therefore combine this

[4]These result valid for a manipulator modeled without friction. The only assumption made in the paper is that the torque limits have a dependency on the joint velocities $\dot{\mathbf{q}}^i$ that is at most quadratic in them.

with (15) and write the combined torque, acceleration and velocity constraints in the form:

$$\begin{array}{l} 0 \leq \dot{s} \leq \dot{s}_{max}(s) \\ \alpha_{min}(s,\dot{s}) \leq \frac{d\dot{s}}{ds} \leq \alpha_{max}(s,\dot{s})). \text{ Where,} \\ 0 \leq \dot{s} \leq \dot{s}_{max}(s) \quad \Rightarrow \quad \alpha_{min}(s,\dot{s}) \leq \alpha_{max}(s,\dot{s}) \end{array} \tag{16}$$

For the case in which the performance index is minimum total travel time, we have reformulated the into the simpler Decoupled Minimum-Time Time-Parameterization problem (DMTTP).

DMTTP problem: Given the functions $\alpha_{min}(s,\dot{s})$, $\alpha_{max}(s,\dot{s})$ and $\dot{s}_{max}(s)$ such that the following properties hold:

$$\begin{array}{l} \forall s \in [s_0, s_f] \dot{s}_{max}(s) \geq 0 \\ \forall s \in [s_0, s_f] \dot{s} \leq \dot{s}_{max}(s) \quad \Rightarrow \quad \alpha_{min}(s,\dot{s}) \leq \alpha_{max}(s,\dot{s}) \end{array}$$

Find the time parameterization $\dot{s}(s)$ for $s \in [s_0, s_f]$ that minimizes the performance index $\mathcal{J} = t_f = \int_{s_0}^{s_f} \frac{1}{\dot{s}(s)} ds$ subject to the constraints:

$$\begin{array}{l} \forall s \in [s_0, s_f] \dot{s}(s) \leq \dot{s}_{max}(s) \\ \forall s \in [s_0, s_f] \alpha_{min}(s,\dot{s}) \leq \frac{d\dot{s}}{ds} \leq \alpha_{max}(s,\dot{s}) \end{array}$$

There are some useful results associated with the DMTTP problem. The following useful lemma is proved in [6].

Lemma 1 *Let $\{[s_0, s_f], \dot{s}(s_0), \dot{s}(s_f), \alpha_{min}, \alpha_{max}, \dot{s}_{max}\}$ be an instance on the DMTTP problem and $\dot{s}^*(s)$ be its solution. Then for any function $\dot{s}(s)$ that satisfies the constraints of the problem, we have $\dot{s}(s) \leq \dot{s}^*(s)$, $\forall s \in [s_0, s_f]$.*

This lemma and the results that preceded it are used to prove that specific phase-space integration schemes will yield the optimal trajectory. In particular the algorithms presented in [6, 16, 15], solve the DMTTP problem.

3 Proximate DMTTP problem

In this section we present the main contribution of this paper: a non-iterative $\mathcal{O}(N_{dof})$ algorithm to find a proximate-optimal solution to the DMTTP problem. This is achieved by transforming the DMTTP problem into one with stricter constraints. We then find the optimal solution to this modified problem. In many cases these stricter constraints will result in trajectories that aren't significantly slower. Further research is needed to precisely characterize the performance loss with respect to the true time-optimal path. We will be addressing this pressing issue in a future paper.

The key idea to obtain an $\mathcal{O}(L)$ algorithm is to break the phase-space integration into several independent sections. In addition, the resulting algorithm is highly parallel. An instance of the DMTTP problem (and hence its solution $\dot{s}^*(s)$, is completely determined by the boundary conditions $\{\dot{s}(s_0), \dot{s}(s_f)\}$ and the constraint functions $\{\alpha_{min}(s,\dot{s}), \alpha_{max}(s,\dot{s}), \dot{s}_{max}(s)\}$. If we knew in advance any point in the optimal path $\{s_d, \dot{s}^*(s_d)\}$ with $s_0 < s_d < s_f$, we could break the problem into two independent ones: first solve for $s_0 \leq s \leq s_d$ and then for $s_d \leq s \leq s_f$. The value $\dot{s}^*(s_d)$ provides the boundary condition at $s = s_d$ that allows the problem to be divided. Any point along the optimal phase-space trajectory, $\dot{s} = \dot{s}^*(s)$ can be used in this manner. Figure 1 shows one such phase-space trajectory. The algorithm is based in the following characterization of a subset of the points in the optimal trajectory. This characterization allows their early identification.

Theorem 1 *Let* $\{[s_0, s_f], \dot{s}(s_0), \dot{s}(s_f), \alpha_{min}(s,\dot{s}), \alpha_{max}(s,\dot{s}), \dot{s}_{max}(s)\}$ *be an instance of the DMTTP problem with the following property:*

$$\begin{array}{l} \forall s \in [s_0, s_f] : \\ \dot{s} \leq \dot{s}_{max}(s) \;\;\Rightarrow\;\; \alpha_{min}(s,\dot{s}) \leq 0 \leq \alpha_{max}(s,\dot{s}) \end{array} \tag{17}$$

Then the following holds:

$$\begin{array}{l} \forall s_1, s_2, s_d \in [s_0, s_f] : \\ \left.\begin{array}{l} s_1 < s_d < s_2 \\ \dot{s}_{max}(s_d) = \min_{s_1 \leq s \leq s_2}\{\dot{s}_{max}(s)\} \\ \dot{s}^*(s_1) = \dot{s}_{max}(s_d) = \dot{s}^*(s_2) \end{array}\right\} \;\Rightarrow\; \begin{array}{l} \dot{s}^*(s_d) = \\ = \dot{s}_{max}(s_d) \end{array} \end{array}$$

Proof. It suffices to show that the phase-space trajectory defined by

$$\dot{s}(s) = u(s) \overset{\text{def}}{=} \begin{cases} \dot{s}_{max}(s_d) & \text{for } s_1 < s < s_2 \\ \dot{s}^*(s) & \text{otherwise} \end{cases}$$

Satisfies all the constraints. For once this is proven, applying lemma 1, we see that $\dot{s}^*(s_d) \geq u(s_d) = \dot{s}_{max}(s_d)$ which combined with the constraint $\dot{s}^*(s_d) \leq \dot{s}_{max}(s_d)$ imply $\dot{s}^*(s_d) = \dot{s}_{max}(s_d)$.

In view of its definition, we only need to show the $u(s)$ satisfies the constraints in the interval $]s_1, s_2[$. Now in this interval our hypothesis guarantee $\dot{s}(s) = \dot{s}(s_d) \leq \dot{s}_{max}(s)$ and since $\dot{s}(s)$ is constant, $\frac{d\dot{s}}{ds} = 0$ and therefore $\alpha_{min}(s,\dot{s}) \leq 0 = \frac{d\dot{s}}{ds} \leq \alpha_{max}(s,\dot{s})$.

Corollary 1 *The theorem holds even if we relax the equality* $\dot{s}^*(s_1) = \dot{s}_{max}(s_d) = \dot{s}^*(s_2)$ *to* $\dot{s}^*(s_1) \geq \dot{s}_{max}(s_d) \leq \dot{s}^*(s_2)$

Proof. Since $\dot{s}^*(s)$ is continuous and $\dot{s}^*(s_d) \leq \dot{s}_{max}(s_d)$, the intermediate value theorem guarantees that there are values $\tilde{s_1}, \tilde{s_2}$ such that $s_1 \leq \tilde{s_1} \leq s_d \leq \tilde{s_2} \leq s_2$ with $\dot{s}^*(\tilde{s_1}) = \dot{s}_{max}(s_d) = \dot{s}^*(\tilde{s_2})$. We can now apply the theorem to $\tilde{s_1}, \tilde{s_2}, s_d$.

The above theorem and its corollary provide a sufficient condition for a phase-space point $\{s, \dot{s}_{max}(s)\}$ to belong to the optimal phase-space trajectory $\dot{s}^*(s)$. This characterization only applies when we can guarantee that the pre-condition $\{\dot{s} \leq \dot{s}_{max}(s) \;\;\Rightarrow\;\; \alpha_{min}(s,\dot{s}) \leq 0 \leq \alpha_{max}(s,\dot{s})\}$ holds. Notice that this condition can always be enforced by redefining $\dot{s}_{max}(s)$ to be

$$\dot{s}_{max}(s) \leftarrow \min \left\{ \begin{array}{l} \dot{s}_{max}(s) \\ \max\{\dot{s} \mid \alpha_{min}(s,\dot{s}) \leq 0 \leq \alpha_{max}(s,\dot{s})\}\} \end{array} \right.$$

This new $\dot{s}_{max}$ is the more strict limit that we were referring to.

The fact that the optimal trajectory touches the boundary curve $\dot{s} = \dot{s}_{max}(s)$ at a finite number of points is also key to the algorithms presented in [6, 16, 18]. Reference [16] shows that for the case in which there are no limits in $\dot{\mathbf{q}}$ (and therefore the boundary $\dot{s}_{max}(s)$ is given by the equation $\alpha_{min}(s,\dot{s}) = \alpha_{max}(s,\dot{s})$) the "switching points" satisfy the necessary condition $\frac{d\dot{s}_{max}(s)}{ds} = \alpha_{min}(s, \dot{s}_{max}(s))$. The algorithm presented in [18] exhaustively classifies these points and presents an efficient method to calculate them.

Our approach differs from the above in that thanks to the introduction of the extra requirement: $\{\dot{s} \leq \dot{s}_{max}(s) \;\;\Rightarrow\;\; \alpha_{min}(s,\dot{s}) \leq 0 \leq \alpha_{max}(s,\dot{s})\}$ we have obtained a *sufficient* condition. This is key to reducing the algorithmic complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$ as discussed in sections 4.4 and 4.5.

4 Algorithm

We present the algorithm and prove its correctness in the continuous domain, and leave its discrete implementation for section 4.3.

4.1 Continuous-Domain version

Assume an instance of the DMTTP problem $\{[s_0, s_f], \dot{s}(s_0), \dot{s}(s_f), \alpha_{min}(s, \dot{s}), \alpha_{max}(s, \dot{s}), \dot{s}_{max}(s)\}$ that satisfies the requirement (17) in theorem 1.

The algorithm to integrate $\dot{s} = \dot{s}_a(s)$ consists of three steps:

1. **Initialization**. Let

$$\mathcal{H} = \{s_{pd} \in [s_0, s_f] \;/\; \dot{s}_{max}(s_{pd}) \text{ is a local minima}\}$$
$$\text{And, } s_l \leftarrow s_0,\ \dot{s}_l \leftarrow \dot{s}(s_0),\ s_r \leftarrow s_f,\ \dot{s}_r \leftarrow \dot{s}(s_f)$$

2. **Integration**. Let

$$\begin{array}{ll} s_1 & \leftarrow s_l,\ \dot{s}_1 \leftarrow \dot{s}_l,\ s_2 \leftarrow s_r,\ \dot{s}_2 \leftarrow \dot{s}_r \\ \mathcal{H} & = \mathcal{H} \cap\,]s_l, s_r[\\ s_d & \leftarrow \{s_{lm} \in \mathcal{H} \text{ s.t. } \dot{s}_{max}(s_{lm}) = \min_{s \in \mathcal{H}}\{\dot{s}_{max}(s)\}\} \\ \dot{s}_d & \leftarrow \dot{s}_{max}(s_d) \end{array}$$

   While ($[s_1 < s_2] \wedge [(\dot{s}_a(s_1) < \dot{s}_d) \vee (\dot{s}_a(s_2) < \dot{s}_d)]$ **Do :**
   If $\dot{s}_a(s_1) \leq \dot{s}_a(s_2)$ integrate forward (i.e. increasing s_1) along the maximum acceleration curve:

$$\frac{d\dot{s}_a}{ds}(s_1) = \left\{ \begin{array}{l} \alpha_{max}(s_1, \dot{s}_a(s_1)) \textbf{ if } \dot{s}_a(s_1) < \dot{s}_{max}(s_1) \\ \min\{\frac{d\dot{s}_{max}}{ds}(s_1), \alpha_{max}(s_1, \dot{s}_a(s_1)\ \}\textbf{else} \end{array} \right.$$

   Else $\dot{s}_a(s_1) > \dot{s}_a(s_2)$ and we integrate backward (decreasing s_2) along the maximum deceleration curve:

$$\frac{d\dot{s}_a}{ds}(s_2) = \left\{ \begin{array}{l} \alpha_{min}(s_2, \dot{s}_a(s_2)) \textbf{ if } \dot{s}_a(s_1) < \dot{s}_{max}(s_2) \\ \max\{\frac{d\dot{s}_{max}}{ds}(s_2), \alpha_{min}(s_2, \dot{s}_a(s_2))\ \}\textbf{else} \end{array} \right.$$

   At any point in the integration, if any element of $\mathcal{H}$ falls outside the (changing) interval $]s_1, s_2[$, we eliminate it from $\mathcal{H}$ and recompute s_d if required.

   The condition $s_1 = s_2$ indicates the integration between s_l and s_r has completed and we return. The condition $(\dot{s}_a(s_1) \geq \dot{s}_d) \wedge (\dot{s}_a(s_2) \geq \dot{s}_d))$ indicates we can break the problem into two independent ones and we continue with the next step.

3. **Separation**. Here we know that $(\dot{s}_a(s_1) \geq \dot{s}_d) \wedge (\dot{s}_a(s_2) \geq \dot{s}_d))$. We divide the problem into the following two:

   i) $s_l \leftarrow s_1,\ \dot{s}_l \leftarrow \dot{s}_a(s_1),\ s_r \leftarrow s_d,\ \dot{s}_r \leftarrow \dot{s}_d = \dot{s}_{max}(s_d)$

   ii) $s_l \leftarrow s_d,\ \dot{s}_l \leftarrow \dot{s}_d = \dot{s}_{max}(s_d),\ s_r \leftarrow s_2,\ \dot{s}_r \leftarrow \dot{s}_a(s_2)$

   And then invoke (recursively) the integration step on each one of the subproblems.

We can see these steps exercised in Figure 1. Each local minimum of $\dot{s}$ served as a decoupling point at some point during the integration.

4.2 Algorithm correctness

In this section we prove that the algorithm integrates the optimal phase-space trajectory i.e. $\dot{s}_a(s) = \dot{s}^*(s)\ \forall s \in [s_0, s_f]$.

It suffices to show that given boundary conditions $(s_l, \dot{s}_l)$ and $(s_r, \dot{s}_r)$ in the optimal trajectory, the integration step always generates points in the optimal trajectory. Once we show this, theorem 1 guarantees that the separation step generates boundary conditions in the optimal path.

It is clear by construction that $\dot{s}_a(s) \geq \dot{s}'(s)$ for any function $\dot{s}'(s)$ that satisfies the constraints and the boundary conditions. In view of lemma 1 all we have to prove is that $\dot{s}_a(s)$ itself satisfies the constraints.

Clearly by construction $\dot{s}_a(s) \leq \dot{s}_{max}(s)\ \forall s \in [s_l, s_r]$, so this constraint is always satisfied. We will prove by contradiction that the slope constraints are satisfied. Let $s_v \in [s_l, s_r]$ be the first value of s that violates the slope constraints. Given the way we choose $\frac{d\dot{s}_a}{ds}(s)$ and the fact that $\dot{s}_a(s_v) \leq \dot{s}_{max}(s_v) \Rightarrow \alpha_{min}(s_v, \dot{s}_a(s_v)) \leq \alpha_{min}(s_v, \dot{s}_a(s_v))$, the only way the slope constraint could be violated is when $\dot{s}_a(s_v) = \dot{s}_{max}(s_v)$. In this case we will violate the slope constraint (16) if either during forward integration we have

$$s_1 = s_v \text{ and } \frac{d\dot{s}_a}{ds}(s_v) = \frac{d\dot{s}_{max}}{ds}(s_v) < \alpha_{min}(s_v, \dot{s}_a(s_v))$$

or during backward integration:

$$s_2 = s_v \text{ and } \frac{d\dot{s}_a}{ds}(s_v) = \frac{d\dot{s}_{max}}{ds}(s_v) > \alpha_{max}(s_v, \dot{s}_a(s_v))$$

We will show that this is impossible for the first (forward integration) case, the backward integration case being analogous. First we must realize that the terminating conditions of the integration step and our selection of $\mathcal{H}$ and s_d guarantee that the invariant $\forall s \in]s_1, s_2[\ : \dot{s}_a(s_1) \leq \dot{s}_{max}(s) \leq \dot{s}_a(s_2)$ holds at all times during the integration. Then it is obvious that $\dot{s}_a(s_v) = \dot{s}_{max}(s_v)$, $s_1 < s_2$ and $\frac{d\dot{s}_{max}}{ds}(s_v) < \alpha_{min}(s_v, \dot{s}_a(s_v)) \leq 0$ violate this invariant. This contradicts our assumptions and therefore, there is no point s_v where our constraint is violated.

4.3 Discrete Implementation

This section presents how the algorithm is adapted to the fact that our input is a discrete sequence of via points $\{\mathbf{q}[k]\}_{k=0}^{N}$ (and not a continuous and differentiable path). The "standard" way to handle this situation would first fit a smooth curve through the via points, obtaining a continuous curve $\mathbf{q} = \mathbf{f}(s)$ it would then apply the continuous version of the algorithm to obtain $\dot{s}(s)$; from this we can integrate $t(s) = \int_0^s \frac{ds'}{\dot{s}(s')}$ and (numerically) invert the function to obtain $s(t_k)$. At the times of interest t_k, we can use $s(t_k)$, $\dot{s}_a(s(t_k))$ in combination with $\mathbf{f}_s$, $\mathbf{f}_{ss}$ and equations (1) and (2) to obtain whichever values of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ and/or $\boldsymbol{\tau}$ are required by the controller.

This procedure, has drawbacks in the context of on-line trajectory generation. It is computationally expensive, requiring $s(t_k)$ to be inverted at every sample. Requires high communication bandwidth between the time-parameterization and controller modules (the desired state must be produced at each sample time). And, it isn't suited for the case in which the sample rate isn't known to the time-parameterization. For these reasons, we have adopted a different approach.

We have based our approach on the observation that, in most cases, the via points are fairly sparse compared to the motion of the robot between samples. Via points are often generated in a "conservative" manner in the sense that portions of the path that are geometrically complicated (e.g. avoiding some obstacle) require for their description, a greater density of via points per unit of path length. In our method, the first few steps are similar to the prototype "standard" method presented above, but the remaining is significantly different:

1. Approximate the path length at each via point.

$$s[0] = 0\ ; s[k+1] = s[k] + \|\mathbf{q}[k]\|\ ,\ \text{for } k = 0...N$$

2. Fit a third order spline interpolation $\mathbf{q} = \mathbf{f}(s)$ to the sequence of points $\{(s[k], \mathbf{q}[k])\}_{k=0}^{N}$

3. Use the spline interpolation to obtain $\dot{s}_{max}[k]$, $\alpha_{min}[k]$, $\alpha_{max}[k]$. Assume the values of the constraints at intermediate points are obtained by linear interpolation.

4. To ensure that $\dot{s}_a(s)$ is continuous, we assume for $s[k] \leq s \leq s[k+1]$, $k = 0..N$ the functional dependency:

$$\frac{d\dot{s}_a}{ds}(s) = \frac{d\dot{s}_a}{ds}(s[k]) + \beta[k]\frac{s - s[k]}{s[k+1] - s[k]}$$

   The integration step will find a value of $\beta[k]$ that approximates the maximum acceleration (deceleration) curve without violating the constraints. To guarantee that $\beta[k]$ can always be found, we impose an extra constraint at each integration step. Namely given $(s[k], \frac{d\dot{s}_a}{ds}(s[k]))$ we require from $\beta[k]$ not only that is satisfies the usual constraints (16) but also that the resulting state at the next via point $(s[k+1], \frac{d\dot{s}_a}{ds}(s[k+1]))$ is such that $\beta[k+1] = -\frac{d\dot{s}_a}{ds}(s[k+1])$ satisfies the constraints in the interval $[s[k+1], s[k+2]]$. This is extra constraint is the price to get a continuous $\frac{d\dot{s}_a}{ds}$.

5. Once the values of $\beta[k]$ and hence $\dot{s}_a[k] \overset{\text{def}}{=} \dot{s}_a(s[k])$ and $\alpha[k] \overset{\text{def}}{=} \frac{d\dot{s}_a}{ds}(s[k])$ are obtained we obtain the via times by simple integration:

$$\Delta t[k] \overset{\text{def}}{=} \int_{s[k]}^{s[k+1]} \frac{ds}{\dot{s}_a(s)} =$$
$$\int_{s[k]}^{s[k+1]} \frac{ds}{\dot{s}_a[k] + \alpha[k](s - s[k]) + \frac{1}{2}\beta[k]\frac{(s-s[k])^2}{s[k+1]-s[k]}}$$

   This integral can be computed analytically.

6. Now that the via times $\{t[k]\}_{k=0}^{N}$ are known, we fit third order splines through the points $\{(t[k], \mathbf{q}[k]\}_{k=0}^{N}$. This involves one spline interpolation per degree of freedom.

4.4 Complexity of the algorithm

The algorithm presented (and its discrete implementation) have running times that are proportional to the length of the path (i.e it is $\mathcal{O}(L)$). Figure 2 shows the execution-time dependency with the number of via points and degrees of freedom. Appendix A for a description of the sequences of via points used in this plots.

The following argument shows that the time-complexity is $\mathcal{O}(N_{via} \times N_{dof})$, N_{via} being the number of via points and N_{dof} being the number of degrees of freedom (DOF). Each operation in the algorithm can be associated with the via point being integrated at that time. Since each via point is integrated just once and, there is a constant number of such operations per integration step the integration is $\mathcal{O}(N_{via})$. The remaining steps are also $\mathcal{O}(N_{via} \times N_{dof})$:

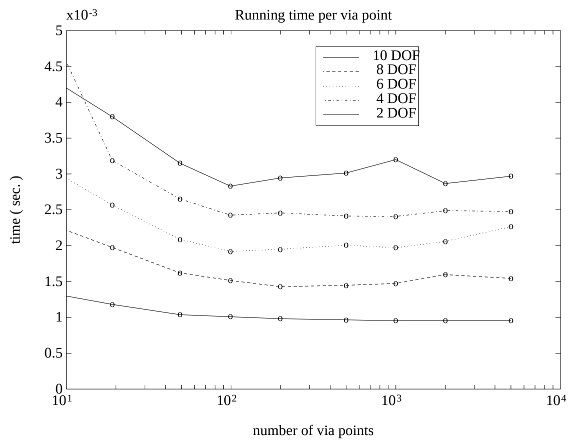

Figure 2: **Running time versus number of via points for different number of degrees of freedom**

This figure illustrates that the time-complexity of the algorithm is linear with the number of via points (or path length). We show that the execution time per via point is approximately constant over a 3 order of magnitude change in the number of via points. The timing corresponds to a Sparc station 2.

- The precomputation stage (steps 1, 2 and 3) of the discrete algorithm are clearly $\mathcal{O}(N_{via} \times N_{dof})$. In particular each spline interpolation for each DOF is an $\mathcal{O}(N_{via})$ operation.
- The integration (step 4) is a $\mathcal{O}(N_{via})$ operation as we have justified before.
- Computing the via times by integrating $1/\dot{s}$ (step 5) is clearly an $\mathcal{O}(N_{via})$ operation.
- Fitting the final splines (step 6) is an $\mathcal{O}(N_{via} \times N_{dof})$ operation, each spline (to each DOF) being $\mathcal{O}(N_{via})$.

Figure 2 corroborates this analysis.

4.5 Complexity of other approaches

In the previous section we have shown the worst-case complexity of our algorithm to be $\mathcal{O}(L) \times N_{dof})$ where L is the length of the path. In this section we will discuss the worst-case complexity of other approaches in the literature. Obviously worst-case complexity is a very crude measure of the practical performance of an algorithm. Not only it measures asymptotic behavior which may represent "pathological" scenarios that may not arise in practice, but it also neglects the constant and lower-order coefficients that may be the most relevant in realistic situations. The real performance of different optimal and pseudo-optimal time-parameterization algorithms in practical paths should be determined using statistical methods by comparing the performance of different algorithms in different sets of paths. To this end, the canonical paths introduced in appendix A may prove useful. This will the topic of a future paper. Worst-case complexity is nevertheless a useful metric in that it gives an upper bound which may be very useful for on-line applications.

We will not be able to address the worst-case complexity of many approaches that have been proposed in the literature. In the following we will give a brief justification of table 1 for some of the most characteristic algorithms.

Dynamic programming approaches have been described in [14] and [17]. *Shin et. al.* [14] show that this approach has a computational complexity of $\mathcal{O}(L \times N_{\mu}^{2})$ where N_{μ} is the number of points in which they discretize the possible values of $\dot{s}$. Aside from the large space requirements of dynamic programming algorithms, there is a further complication derived from the need to estimate adequate values N_{μ} which may be problem and path-specific. *Singh et. al.* [17] propose a recursive refinement of N_{μ} to address the problem of its selection.

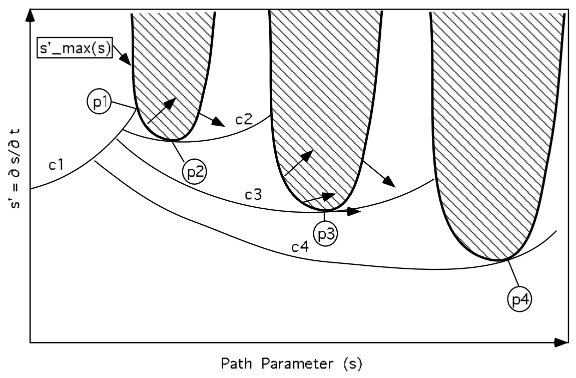

Figure 3: **Worst-case scenario for direct-integration algorithms.**

The algorithm used to obtain the optimal trajectory involves forward integration in phase space along the maximum acceleration curve. If this curve (c_1 above) leaves the admissible region of phase space (point p_1 above), the boundary curve of the admissible region (curve $\dot{s} = \dot{s}_{max}(s)$ is searched for a point where its slope $\frac{d\dot{s}_{max}}{ds}$ matches the slope of the "closed wedge" described in figure 1 (point p_2 above). Starting at p_2 we integrate backward along the maximum decelerating trajectory until the curve it intersects c_1 and forward as before (this generates curve c_2). This process is repeated to generate curves c_3 and c_4. Notice that in this worst-case scenario, each time we have to integrate backwards all the way to the initial curve c_1 (that is, c_3 does not intersect c_2, c_3 does not intersect c_2 nor c_3, etc. Therefore, since the backward integration takes time proportional to the path length L and the number of c_i curves is also proportional to L the running time is $\mathcal{O}(L^2)$.

The algorithms presented by *Bobrow et. al.* [6] and *Shin et. al.* [13] use a direct integration approach. These papers don't address the computational complexity of the algorithms but it can be shown with an example that they have worst-case complexities of $\mathcal{O}(L^2)$. Figure 3 illustrates the general features of a worst-case scenario with running time $\mathcal{O}(L^2)$ for Shin's algorithm. Appendix B) gives a simple analytical path where the scenario described in figure 3 actually occurs. Similar examples can be found for Bobrow's algorithm.

The proximate-time optimal algorithm presented in *Butler et. al.* [2] uses a simplified constraint in place of individual joint-torque constraints which speeds up the computations but direct integration is still used to solve so that the worst-case complexity is still the same. *Slotine et. al.* [18] an analytical characterization of the switching points to obtain drastically more efficient algorithm. The worst-case complexity remains the same.

Shin et. al. [15] also propose an iterative approach, combination of a gradient and binary search techniques. Starting from a path that satisfies all the constraints, each iteration brings the path closer to the optimal by increasing the intermediate velocities $\dot{s}$ whenever this is compatible with all the constraints. The authors show in the paper that the complexity of their algorithm is $\mathcal{O}(N_\lambda^2)$. Where λ is the number of points in which the trajectory parameter s is discretized. This result however refers to the complexity with respect to using smaller step sizes in s for the *same* path. The complexity with respect to path length (i.e. keeping the step-size in s constant but increasing the length of the path) is clearly $\mathcal{O}(L \times N_{iter})$. Where N_{iter} is the number of iterations required by the algorithm to converge. N_{iter} should be reasonably independent from the path length but depends on how close we desire the final approximation to the true optimal path.

4.6 Case of Configuration-Independent Limits in velocities and accelerations

In this section we discuss in the specific case of configuration-independent limits in the velocities and accelerations. These constraints may arise naturally for some problems. For example as pointed by [8], our path may be given in Cartesian space and be limited not by actuator capabilities but by interactions with other objects (they use the example of moving a liquid in an open container as a case in which these constraints would be natural). There are many cases in which our constraints can be approximated in this way without significant performance loss (e.g mobile robots).

Our general approach is, of course, applicable to this specific case. Nevertheless, we have paid special attention to this case and making some conservative approximations, we have developed a computationally more efficient solution.

The approximation consists in replacing the acceleration constraint: $-\ddot{\mathbf{q}}_{max} \leq \ddot{\mathbf{q}} \leq \ddot{\mathbf{q}}_{max} \iff \left|\frac{\ddot{\mathbf{q}}^i}{\ddot{\mathbf{q}}^i_{max}}\right| \leq 1\ ,\ i = 1..n$ by the approximate constraint $\sum_{i=1}^{n}(\frac{\ddot{\mathbf{q}}^i}{\ddot{\mathbf{q}}^i_{max}})^2 \leq 1$. It is useful to write this constraint using a metric tensor $\mathbf{Q}$.

$$\mathbf{Q} \stackrel{\text{def}}{=} \mathbf{diag}[..,\ \frac{1}{\ddot{\mathbf{q}}^i_{max}}\ ,..]$$
$$\|\mathbf{x}\|_Q \stackrel{\text{def}}{=} \mathbf{x}^T\mathbf{Q}\mathbf{x}$$

The constraint now becomes

$$\|\ddot{\mathbf{q}}\|_Q \leq 1 \tag{18}$$

Obviously our new constraint (18) is more strict than (4.6). A simple geometric interpretation follows: constraint (4.6) requires $\ddot{\mathbf{q}}$ to be inside the parallelepiped with sides intersecting the axis at $\pm\ddot{\mathbf{q}}^i_{max}$. The new constraint (18) replaces this parallelepiped with the ellipsoid with principal axes $\ddot{\mathbf{q}}^i_{max}$. It is clear from this interpretation that the sacrifice in performance won't be great. Due to the lack of space we can't present a comparison of the tradeoff between computation time and trajectory-optimality, but typically we see an order of magnitude reduction in computation time with an travel time that within 30% of the optimal.

The use of constraint (18) allows several simplifications. We start by measuring path-length using the metric tensor $\mathbf{Q}$. That is $ds^2 = d\mathbf{q}^T\mathbf{Q}d\mathbf{q}$. Then we can see that

$$\|\mathbf{f}_s\|_Q = 1$$
$$\mathbf{f}_s{}^T\mathbf{Q}\mathbf{f}_{ss} = 0$$
$$\|\ddot{\mathbf{q}}\|_Q^2 = \left\|\mathbf{f}_s\ddot{s} + \mathbf{f}_{ss}\dot{s}^2\right\|_Q^2 = \ddot{s}^2 + \|\mathbf{f}_{ss}\|_Q^2\,\dot{s}^4$$
$$\|\ddot{\mathbf{q}}\|_Q \leq 1 \iff |\ddot{s}| \leq \ddot{s}_{max}(s,\dot{s})$$
$$\ddot{s}_{max}(s,\dot{s}) \stackrel{\text{def}}{=} \sqrt{1 - \dot{s}^4\,\|\mathbf{f}_{ss}(s)\|_Q^2} = \sqrt{1 - (\frac{\dot{s}}{\dot{s}^a_{max}(s)})^4}$$
$$\dot{s}^a_{max}(s) \stackrel{\text{def}}{=} \frac{1}{\sqrt{\|\mathbf{f}_{ss}(s)\|_Q}}$$

We see that the acceleration constraint imposes the limit $|\dot{s}| \leq \dot{s}^a_{max}(s)$ we can combine this limit by the one imposed by the velocity constraint (15) and write the constraints as:

$$|\dot{s}| \leq \dot{s}_{max}(s) \overset{\text{def}}{=} \min\{\dot{s}^a_{max}(s), \gamma(s)\} \tag{19}$$

$$|\ddot{s}| \leq \ddot{s}_{max}(s, \dot{s}) \iff \left|\frac{d\dot{s}}{ds}\right| \leq \alpha_{max}(s, \dot{s}) \tag{20}$$

$$\alpha_{min}(s, \dot{s}) = \ddot{s}_{min}(s, \dot{s})/\dot{s} \text{ For } \dot{s} \neq 0 \tag{21}$$

Again we have reduced the constraints to the general form of an instance of the DMTTP problem. The difference is that now $\dot{s}_{max}(s)$ and $\alpha_{min}(s, \dot{s})$ are very easy (and fast) to compute. Also we note that $\dot{s} \leq \dot{s}_{max}(s) \leq \dot{s}^a_{max}(s) \quad \Rightarrow \quad \alpha(s, \dot{s}) \geq 0$ to theorem 1 applies without any further limitations on $\dot{s}_{max}(s)$.

Section 5 shows the performance of this algorithm on several "canonical" paths.

5 Results

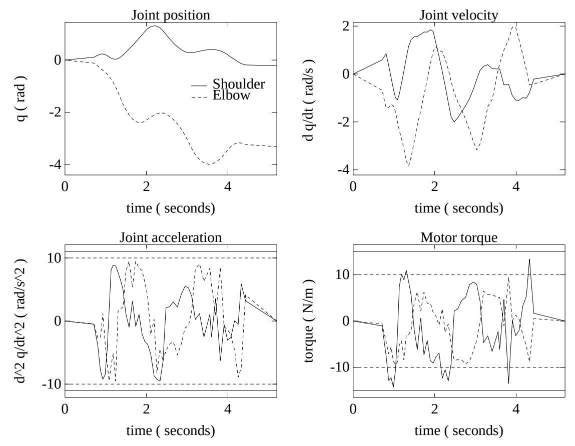

Figure 4: **Optimal time-parameterization for a 2 DOF scara-type manipulator with acceleration and torque constraints.**

Result of time-parameterizing a set of joint-space via points for a 2 DOF scara-type manipulator subject to configuration-independent acceleration and torque constraints. We can observe that both accelerations and torques remain within their limits.

In this section we present the results of time-parameterizing some sample sets of via points. Figure 4 shows the proximate optimal path for a scara-type two degree-of-freedom (DOF) manipulator subject to joint-space acceleration and torque limits. This results correspond to the phase-space trajectory shown in figure 1. The dynamics of this manipulator are those of the Dual Arm Flexible Drive-Train Robot at the Aerospace Robotics Laboratory . These arms are described in [10]. We can observe that both the accelerations and the torques are within the limits. We also observe the typical bang-bang trajectories resulting from the optimization with inequality constraints.

Figure 5 shows results for a 6 DOF manipulator with constraints in the maximum joint-space acceleration. For simplicity we have not used torque constraints and therefore no dynamic equations are necessary. The 6 DOF joint-space path has been generated with the technique described in the Appendix A.

6 Conclusions

We have described an efficient algorithm to obtain a continuous trajectory that traverses a sequence of via points. The trajectory is proximate time-optimal subject to some dynamic constraints which can be any combination of velocity, acceleration or torque limits which may be configuration dependent. The algorithm has time-complexity $\mathcal{O}(N)$ in the number of via points. It is therefore suitable for on-line trajectory generation. Simulation results (See figure 2) using the "canonical" sequences of via points described in appendix A show typical performances of 20 ms per via point for a 6 DOF manipulator [5]. We are currently using the algorithm in the Dual Arm Flexible Drive-

[5]This timing corresponds to a Sun Sparc station 2

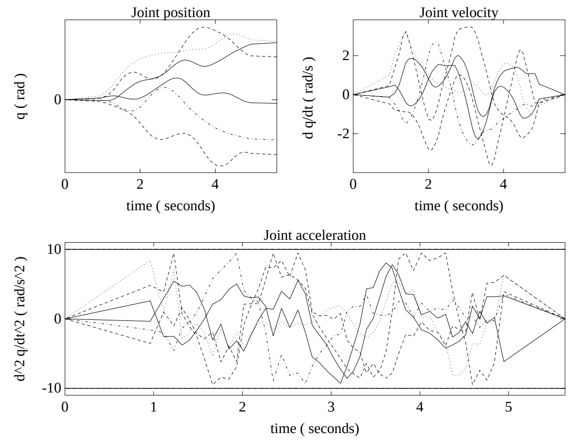

Figure 5: **Optimal time-parameterization for a 6 DOF manipulator with acceleration constraints.**

Result of time-parameterization of a set of joint-space via points for a 6 DOF manipulator subject to configuration-independent acceleration constraints. To facilitate the visual inspection we have used the same acceleration limits for each degree of freedom.

Train Robot at the Aerospace Robotics Laboratory [9] in combination with a randomized planner developed at the Computer Science Robotics Laboratory [20].

A Canonical Paths

To test the algorithm's performance and obtain experimental information on its time-complexity, we need "canonical" paths with different number of via points/DOF. This sequences of via points must be ergodic in their geometrical properties. Our approach has been to generate a long sequence of via points and extract from it sub-sequences of different lengths.

We have used smoothed random walks to generate our sequences of via points. A random walk is inherently ergodic and can be used to generate arbitrarily long sequences of via points in any number of degrees of freedom. Smoothing the random walk brings two benefits: It clusters together via points in a manner proportional to the local curvature and, it produces "smoother" paths that can be traversed at higher speeds by the manipulator without exceeding its dynamic limits. We have used a zero-phase forward and reverse digital low-pass butterworth filter (smoother). The parameters used for the examples in this paper are: A random-walk step of 1 radian in each degree of freedom, and an order N_{dof} low-pass butterworth filter with a spatial cutoff frequency of 0.15.

Figure 6 illustrate 2 DOF paths with increasing number of via points. These are the same paths used throughout the paper. In particular, the first path corresponds to the results in figure 1 and figure 4.

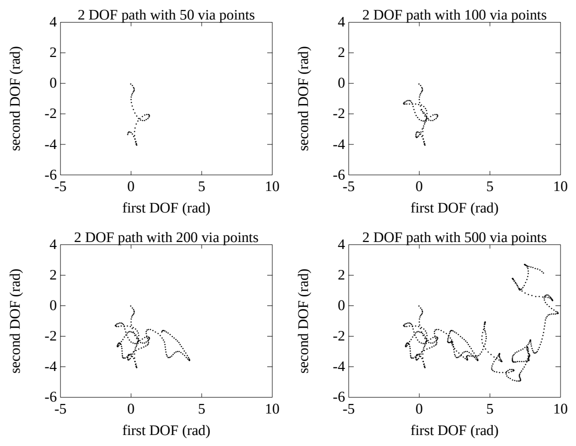

Figure 6: **Filtered random walks of different length for 2 degrees of freedom**

These four sets of via points represent paths of increasing length. These sequences are ergodic in their geometrical properties. We have shown sets of 50, 100, 200 and 500 via points.

B Worst-Case Example

Our goal is to construct a simple, analytically tractable instance of the DOTP problem that exhibits the behavior sketched in figure 3. To this end, we will build the geometric path in 2 dimensions as a piecewise connection of $2N + 1$ circular arcs with varying radii, here N is increased to obtain paths of increasing length:

$$\begin{aligned}
&\text{for} \quad s \in [L_k, L_{k+1}[\quad \text{define} && (22)\\
q_1(s) &= x_k + A_k cos(s/A_k + \phi_k) && (23)\\
q_2(s) &= y_k + A_k sin(s/A_k + \phi(k)) && (24)\\
A_k &\overset{\text{def}}{=} |N - k| + 2 && (25)\\
\phi_k &\overset{\text{def}}{=} \phi_0 + \sum_{i=1}^{k} L_i(1/A_{i-1} - 1/A_i) && (26)\\
x_k &\overset{\text{def}}{=} x_0 + \sum_{i=1}^{k} (A_{i-1} - A_i)\cos(\frac{L_i}{A_i} + \phi_i) && (27)\\
y_k &\overset{\text{def}}{=} y_0 + \sum_{i=1}^{k} (A_{i-1} - A_i)\sin(\frac{L_i}{A_i} + \phi_i) && (28)\\
L_k &\overset{\text{def}}{=} k\pi && (29)\\
& && (30)
\end{aligned}$$

Equations 23 thru 25 define a path composed of N arc pieces of decreasing radii followed by N pieces of increasing radii. Equations 26 thru 28 represent the centers and phase shifts defined

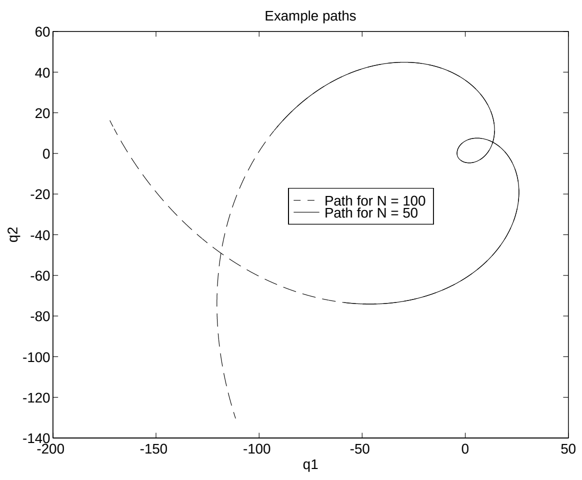

Figure 7: **Two worst-case paths of different lengths. Each paths is composed of $2N$ circular arcs.**

so that the resulting path is continuous and has a continuous tangent. ϕ_0, x_0 and y_0 are chosen so that $q_1(N/2) = q_2(N/2) = 0$. Figure 7 shows two example paths for different values of N. Notice that the path is are symmetrical about the $k = N + 1$ segment.

We will use maximum acceleration $|\ddot{\mathbf{q}}| \leq a \overset{\text{def}}{=} 0.5$ as the constraint. It is easy to show that the constraints 16 reduce to

$$\begin{aligned} 0 &\leq \dot{s} \leq \dot{s}_{max}(s) = \sqrt{aA_k} \overset{\text{def}}{=} \dot{s}_{max}(k) && (31) \\ \left|\frac{d\dot{s}}{ds}\right| &\leq \frac{a}{\dot{s}}\sqrt{1 - \frac{\dot{s}^4}{\dot{s}_{max}(k)^4}}, \quad s \in [L_k, L_{k+1}] && (32) \end{aligned}$$

For a given $N >= N_0$, we will now show that starting from $\dot{s} = 0$ the maximum acceleration curve $\dot{s} = \dot{s}_{acc}(s)$, will leave the admissible region of phase space limited by the curve $\dot{s} = \dot{s}_{max}(s)$ at a point $s_0(N) \leq L_{N/2}$. This can be proven by contradiction, assuming $s_0(N) > L_{N/2}$ we would have in the interval $0 \leq s \leq L_{N/4}$, $0 \leq k \leq N/4$:

$$\begin{aligned} \dot{s}_{acc}(s) &\leq \dot{s}_{max}(N/2) = \sqrt{N/2 + 2} && (33) \\ \dot{s}_{max}(k) &< \dot{s}_{max}(N/4) && (34) \\ \frac{d\dot{s}_{acc}}{ds} &> \frac{a}{\dot{s}_{acc}}\sqrt{1 - \frac{\dot{s}_{max}(N/2)^4}{\dot{s}_{max}(N/4)^4}} && (35) \\ \frac{d\dot{s}_{acc}}{ds} &> \frac{P(N)}{\dot{s}} \Rightarrow \dot{s}_{acc}(s) \geq w(s) && (36) \\ \text{where} \quad w(s) &\ \text{verifies } \frac{dw}{ds} = P(N)/w && (37) \end{aligned}$$

$$
\begin{aligned}
w(s) &= \sqrt{2P(N)(s-s_0)+w(s_0)^2} && (38)\\
P(N) &\overset{\text{def}}{=} a\sqrt{1-\frac{(N/2+2)^2}{(3N/4+2)^2}} && (39)\\
w(0)=0 \quad &\Rightarrow \quad w(L_{N/4}) = \sqrt{\frac{N}{2}\pi P(N)} && (40)\\
&= \sqrt{\frac{N}{2}+(\pi P(N)-1)\frac{N}{2}} && (41)
\end{aligned}
$$

Therefore, since $P(N)$ is strictly increasing and for $N = 31, (\pi P(N)N/2 - 1) > 2$ we see that for $N > 31$, $\dot{s}(L_{N/4}) \geq w(L_{N/4}) > \sqrt{N/2+2}$ which contradicts $\dot{s}_{acc}(s) \leq \dot{s}_{max}(N/2)$. Hence, we have proved that for $N > 31 \;\Rightarrow\; s_0(N) \leq L_{N/2}$.

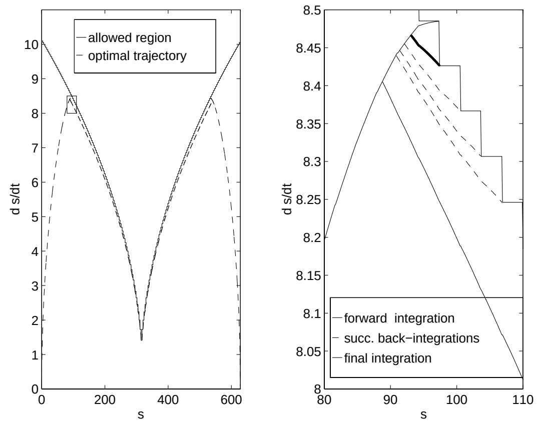

Figure 8: **Phase space direct integration and detail.**

We can see from figure 8 how the situation sketched in figure 3 occurs in this example for $s \geq s_0(N) \in [L_{m(N)}, L_{m(N)+1}[$. Where we have just proven that $m(N) < N/2$. The subsequent forward integration along the maximum acceleration curve will leave the allowed region at each point $s = L_k$ with $m(N) < k < N$. Shin's algorithm [13] then integrates backwards from the point $\dot{s} = \dot{s}_{max}(k+1)$ until it intersects the accelerating curve. However, we will show below that for most of these points (specifically $k < r(N) = N - 80$), this integration must proceed all the way back to $s = L_{m(N)+1}$. The reason is shown in figure 8. For $k < r(N)$ Each backward integration from $(s = L_{k+1}, \dot{s} = \dot{s}_{max}(k+1))$ arrives to $s = L_k$ with a value of $\dot{s}$ smaller than $\dot{s}_{max}(k)$ being therefore unable to intersect the previously computed $\dot{s}(s)$ curve in the region in which it is decreasing, i.e. all the way back to the segment $s \in [0, s_0(N)]$. The symmetric situation is replicated in the backward integration branch that starts at the end of the path. Assuming N large enough, the number of backtracks is $(N - m(N)) - (N - r(N)) \geq N/2 - 80$ i.e. $\mathcal{O}(N)$. The computation required for each backtrack is proportional to the length of the path backtracked. Since the length of each segment is constant (π), this length is $\mathcal{O}(N)$ and therefore the complexity is $\mathcal{O}(N^2) = \mathcal{O}(L^2)$.

Finally we prove that (as stated above) for $k < r(N) = N - 80$, each backward integration from $(s, \dot{s}) = (L_{k+1}, \dot{s}_{max}(k+1))$ along the maximum deceleration curve $\dot{s} = \dot{s}_{dec}(s)$, reaches $s = L_k$ with a value $\dot{s} < \dot{s}_{max}(k)$.

For $s \in [L_k, L_{k+1}[$ we have $\dot{s}_{dec}(s) \geq \dot{s}_{max}(k+1)$ and therefore:

$$\begin{aligned}
\frac{d\dot{s}_{dec}(s)}{ds} &\geq \frac{-1}{\dot{s}_{dec}(s)}\sqrt{1 - \frac{\dot{s}_{dec}(s)^4}{\dot{s}_{max}(k)^4}} \\
&\geq \frac{-1}{\dot{s}_{dec}(s)}\sqrt{1 - \frac{\dot{s}_{max}(k+1)^4}{\dot{s}_{max}(k)^4}} = \frac{-Q(N-k)}{\dot{s}_{dec}(s)} \\
Q(x) &\overset{\text{def}}{=} a\sqrt{1 - \frac{(x+1)^2}{(x+2)^2}}
\end{aligned}$$

Using a bound similar to 36 and integrating backwards from L_{k+1}, to L_k we can obtain:

$$\begin{aligned}
\dot{s}_{dec}(s) &< \sqrt{2Q(N-k)(L_{k+1} - s) + \dot{s}_{max}(k+1)^2} \\
\dot{s}_{dec}(L_k) &< \sqrt{2Q(N-k)\pi + \dot{s}_{max}(k+1)^2} \\
&= \sqrt{2Q(N-k)\pi + a(N-k+1)}
\end{aligned}$$

Since $Q(x)$ is strictly decreasing and $2Q(80)\pi < a = 0.5$, we have that $N - k > 18 \quad \Rightarrow$ $\dot{s}_{dec}(L_k) < \sqrt{N-k+2} = \dot{s}_{max}(k+1)$ as stated.

References

[1] J.E. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE Journal of Robotics and Automation*, 4(4):443–451, August 1988.

[2] J. Butler and M. Tomizuka. A suboptimal reference generation technique for robotic manipulators following specified paths. *Transactions of the ASME, Journalof Dynamic Systems, Measurement, and Control*, 114:524–527, September 1992.

[3] Y. Chen, S. Y.-P Chien, and A. A. Desrochers. General structure of time-optimal control of robotic manipulators moving along prescribed paths. *International Journal of Control*, 56(4):767–782, 1992.

[4] Elmer G. Gilbert and Daniel W. Johnson. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, RA-1(1):21–30, September 1985.

[5] B. Langlois J. Barraquand and J.C. Latombe. 'numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:224–241, March 1992.

[6] S. Dubowsky J.E. Bobrow and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, 4(3), Fall 1985.

[7] J.C. Latombe. *Robot Motion Planning.* Kluger Academic Publishers, Boston, MA, 1991.

[8] J.Y.S. Luh and C.S. Lin. Optimum path planning for mechanical manipulators. *Transactions of the ASME*, 102:142–151, June 1981.

[9] Gerardo Pardo-Castellote, Tsai-Yen Li, Yoshihito Koga, Robert H. Cannon Jr., Jean-Claude Latombe, and Stan Schneider. Experimental integration of planning in a distributed control system. In *Preprints of the Third International Symposium on Experimantal Robotics*, Kyoto Japan, October 1993.

[10] L. E. Pfeffer and R. H. Cannon Jr. Experiments with a dual-armed, cooperative, flexible-drivetrain robot system. In *Proceedings of the International Conference on Robotics and Automation*, Atlanta, GA, May 1993. IEEE, IEEE Computer Society.

[11] Z. Shiller and S. Dubowsky. Robot path planning with obstacles, actuator, gripper and payload constraints. *International Journal of Robotics Research*, 8(6), December 1989.

[12] Z. Shiller and S. Dubowsky. On computing the global time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Transactions on Robotics and Automation*, 7(6):785–797, December 1991.

[13] Kang G. Shin and Neil D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, AC-30(6):531–541, June 1985.

[14] Kang G. Shin and Neil D. McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, AC-31(6):491–500, June 1986.

[15] Kang G. Shin and Neil D. McKay. Minimum-time trajectory planning for industrial robots with general torque constraints. In *IEEE International Conference on Robotics and Automation*, pages 412–415, San Francisco, California, April 6-10 1986.

[16] Kang G. Shin and Neil D. McKay. Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automatic Control*, AC-31(6):501–511, June 1986.

[17] S. Singh and M.C. Leu. Optimal trajectory generation for robotic manipulators using dynamic programming. *Transactions of the ASME*, 109:88–96, June 1987.

[18] J. E. Slotine and H. S. Yang. Improving the efficiency of time-optimal path-following algortihms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, April 1989.

[19] J. T. Wen and A. Desrochers. Sub-time-optimal control strategies for robotic manipulators. In *IEEE International Conference on Robotics and Automation*, San Francisco, California, April 6-10 1986.

[20] J.C. Latombe Y. Koga, T. Lastennet and T.Y. Li. Multi-arm manipulation planning. In *9th International Symposium on Automation and Robotics in Construction*, Tokyo, Japan, June 1992.