Trajectory Planning in the Infinity Norm for Linear Control Systems

Magnus Egerstedt*

magnuse@math.kth.se

Division of Optimization and Systems Theory

Royal Institute of Technology

S - 100 44 Stockholm, Sweden

Clyde F. Martin†

martin@math.ttu.edu

Department of Mathematics

Texas Tech University

Lubbock, 79409 Texas

Abstract

When planning trajectories, a reasonable demand on the generated curves is that they interpolate through given points, or given intervals, at given specified times, or that they go through these points or intervals at times lying in specified time windows. This problem is addressed here as an optimal control problem, minimizing the infinity norm of the inputs of a discretized linear control system since the infinity norm minimization problem arises naturally when doing trajectory planning for both controlled switching systems and nonholonomic robotic systems.

1 Introduction

In this article we look at the problem of finding the control that drives a given linear control system between predefined states, with minimal infinity norm. Instead of solving this hard problem analytically, we show how to

*This work was sponsored in part by the Swedish Foundation for Strategic Research through its Centre for Autonomous Systems at KTH.

†This work was supported in part by NASA Grants NAG 2-902 and NAG 2-899 and by NSF Grant DMS 9628558. During the time that this work was done the second author was a guest professor at KTH.

transform it into a linear programming problem that can be solved in a fast and robust way. Since our approach is going to be based on a discretization of the original continuous problem, the result should be used for trajectory planning where a certain property of the path is wanted, rather than for the construction of the actual continuous controller. We illustrate this fact by looking at an example of a car-like robot, whose nonholonomic steering constraint makes it necessary to be able to generate paths with a minimal second derivative. This is done in order to keep the curvature of the path small at all times. We also make a comparison between our method and generalized splines, generated by a L^2-minimization of the control input. We then extend our method, introducing the concept of smoothing splines, so that the generated trajectory just passes closely to to a given set of points, or through a given set of intervals, instead of, as before, through these points, on which we are given linear interpolation conditions, such as constraints on the position, velocity and so on.

We also investigate a somewhat different type of trajectory planning problem, namely the problem of interpolating curves through, not only specified intervals, but through intervals located in time windows instead of at specified times. Solving this problem in order to generate curves that minimize

some cost function is important for two obvious reasons. The first one is that when planning trajectories, for example in air-craft navigation, it is not necessary to be at a certain point at a given time T, but rather that we reach this point at a time somewhere in the interval around T instead. We do not, for example, demand that a plane should be at the point p exactly at time T, but rather that it should be there close to that time.

The second reason for trying to solve this type of problem is that since we are minimizing some cost function, in this article as a function of the control signal and therefore implicitly depending on time, it was our initial belief that this time window approach greatly could reduce the value of the cost function, for instance the energy produced in the transition between the boundary states, by relaxing the exact time interpolation condition. It will be seen, further on, that this intuition was correct, even though we in this article didn't minimize the energy produced, but rather the infinity norm of the control signal.

In this article, we first, in Section 2, look at the fixed time infinity norm optimization problem for linear control systems, when interpolating through given points, and in Section 3, we show how this approach can be generalized, introducing the smoothing spline concept.

We then show, in Section 4, how we can transform the linear system into a bilinear one by adding time as an extra controlled state variable. We then proceed by proposing a way to solve the time-window-interpolation problem by solving a non-convex optimization problem that this transformation gives rise to.

2 Fixed Time Interpolation

We begin by studying linear single input control systems of the form

$$\dot{x}(t) = Ax(t) + bu(t), \tag{1}$$

were A is a $n \times n$ matrix and b is $n \times 1$. The question that need to be answered is how to minimize the control input in the infinity norm so that the system is driven between predefined boundary states in a given time period. This problem can be formulated in terms of finding a control $\hat{u}(\cdot)$ such that the system is driven between the given states $x(0) = x_0$ and $x(T) = x_T$, while interpolating through some given points or intervals. Here, $u(\cdot)$ is the control that has the lowest maximal absolute value $\forall t \in [0, T]$, so what we want to do is to find the control

$$\hat{u}(\cdot) = \min_u(\sup_t\{|u(t)|\}). \tag{2}$$

Finding this $\hat{u}(\cdot)$ is far from trivial [7], and instead of focusing on the extensive task of trying to find an analytical solution to the problem, (in general it is not even clear if there is one), we aim towards finding a robust numerical method with low complexity. The method that we suggest transforms the problem into a linear programming (LP) problem, for which there already are fast and robust algorithms [8]. It must be stressed that we, in this case, can not use the Pontryagin type methods [6] that are used when dealing with L^2-minimization. In these cases we would instead minimize

$$\int_0^T u(t)^2 dt, \tag{3}$$

where we would get generalized splines of different kinds depending on the eigenvalues of the matrix A [3]. However, in our case, this will not do, since we have no guarantee that the control that minimizes the integral in (3) actually minimizes the supremum over all $t \in [0, T]$.

2.1 LP-Formulation

Given a discretization of the investigated time interval $[0, T]$, we can rewrite (2) as

$$\min(\max\{|u_j|, j = 1, \ldots, N-1\}). \tag{4}$$

In order to get a linear cost function, we need to reformulate this and we first observe that minimizing $\max\{|\xi_j|, j = 1, \ldots, k\}$ is a convex minimization problem, equivalent to minimizing η under the constraints $|\xi_j| \leq \eta, \quad j = 1, \ldots, k$, since by making η as small as possible, we automatically make the largest of the ξ_j's as small as possible. The next observation is that the constraint $|\xi| \leq \eta$ is equivalent to $\xi \leq \eta$ and $-\xi \leq \eta$.

Based on these observations, we obtain

$$\begin{array}{ll} \min & z \\ \text{when} & \begin{cases} u_j \leq z \quad j = 1, \ldots, N-1 \\ -u_j \leq z \quad j = 1, \ldots, N-1. \end{cases} \end{array} \tag{5}$$

We now need to incorporate the dynamics of the system in (1) into the LP-formulation, and we chose a simple, straight forward differential approximation

$$\frac{x_{j+1} - x_j}{h_j} = Ax_j + bu_j, \quad j = 1, \ldots, N-1, \tag{6}$$

where h_i is the length of time interval $t_{j+1} - t_j$.

We now define an observation function, c ($n \times 1$–vector), that determines which of the states (or a linear combination of the states) will be subject to the interpolation constraints. This gives us an output, y, from our control

system

$$y_j = c^T x_j, \tag{7}$$

and our interval interpolation conditions become

$$\alpha_i \leq y_i \leq \beta_i, \quad i \in \mathcal{M}, \tag{8}$$

where α_i and β_i are the lower and upper bounds of the interval that we want to interpolate through. Throughout the article we will have $c = (1, 0, \ldots, 0)^T$ but this is not necessary in the general case.

If we want to interpolate through points, we just shrink the intervals so that $\alpha_i = \beta_i$ for those i:s that are of interest. In (8), $\mathcal{M}$ is just defined as

$$\mathcal{M} = \{i \in [1, N] : t_i \text{ is an interpolation time}\}. \tag{9}$$

The discretization of the time axis has to be chosen in such a way that the interpolation times in (9) are in the set of discrete times. If, for instance, t^1 is the first interpolation time, we can chose an equidistant discretization, with the first $p-1$ time steps as

$$\frac{t^1 - t_0}{p}, \quad i = 1, \ldots, p-1. \tag{10}$$

2.2 An Example - Car-Like Robot

One application in which this type of problem arises naturally is in path planning for robots with nonholonomic constraints on their dynamics. Here, we consider a car-like robot, and in order for a trajectory to be feasible, the curvature of the planned trajectory can not be greater than the curvature, κ, of the trajectory generated when driving the car with maximal steering angle.

One way of generating feasible paths, hopefully within the constraints for a car-like robot, is thus by making the second derivative of the curve as small as possible in infinity norm, since

$$\kappa^2 = \frac{f''(x)^2}{\left(1 + f'(x)^2\right)^3} \leq f''(x)^2. \tag{11}$$

The reason why we are this interested in the curvature is that any path that a car can follow need to have $\kappa \leq \sin(\phi_{\max})/L$, where L is the length of the car, and $\phi_{\max}$ is the maximal steering angle [5], [2].

So, if we want to minimize the second derivative of a scalar function in infinity norm, we simply let our system matrices be

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{12}$$

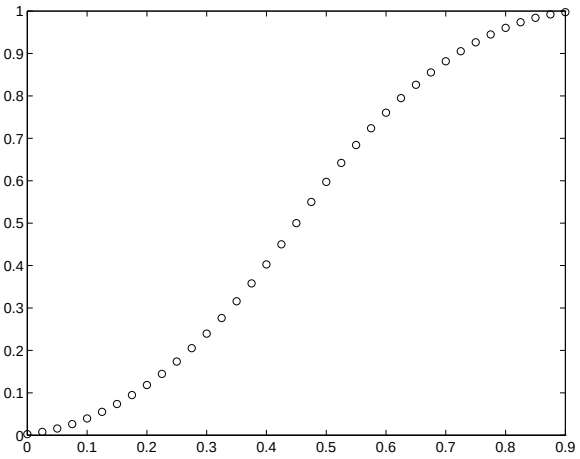


Figure 1: Discretized curve (40 points) with minimal second derivative in the infinity norm.

The resulting curve can be seen in Figure 1, where we have chosen to work with 40 discrete points and boundary conditions

$$x_0 = \begin{pmatrix} 0 \\ \\ 0 \end{pmatrix}, \quad x_T = \begin{pmatrix} 1 \\ \\ 0 \end{pmatrix}. \tag{13}$$

It can also be seen that when comparing with cubic splines, see Figure 2, and our approach did in fact get a much lower value on the maximal second derivative, as seen in Figure 3. Looking closely at this picture gives us that our maximal control value is about 4.2, while the second derivative of the spline goes as high as 7.5. This is actually a large improvement, and for the purpose of generating feasible paths for car-like robots, and other

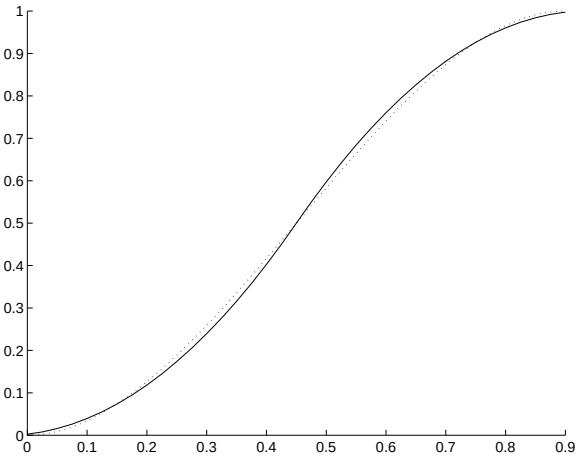


Figure 2: A comparison between our method (solid) interpolated between the discretization point, and the curve generated by cubic splines (dotted).

nonholonomic dynamical systems, our LP method should therefore be of some help and importance when doing trajectory planning for these systems.

If we now want the car to drive through the points $x(0.25) = 0.3$, $x(0.5) = 0.8$ and $x(0.75) = 0.5$, we get the interpolated curve seen in Figure 4. Our method works in this case as well, and since it only uses linear programming, the complexity of the numerical algorithms will not be too high to impose constraints on the method.

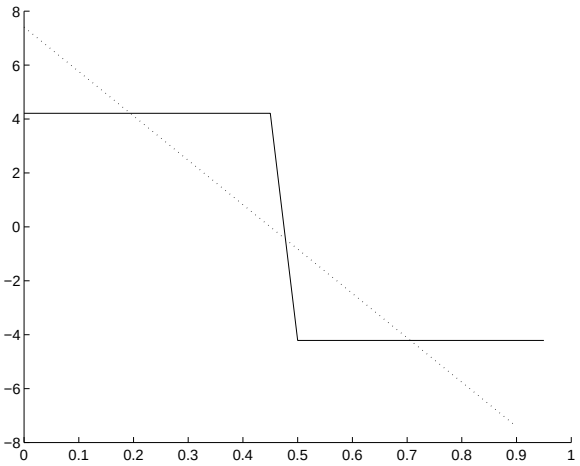


Figure 3: A comparison between the second derivative of our method (solid), that (not too surprisingly) looks much like a Bang-Bang control, and the second derivative of the cubic splines (dotted).

3 The Smoothing Spline Concept

Now that we have a way of constructing solutions to our original problem, we can get inspiration from [13] to take our approach even further. In general, interpolating with splines as well as with our LP-curve, is only reasonable when the data that we are working with is non-noisy. Thus, in a statistical framework it is desirable that one can construct curves that pass near the interpolation points, without being constraint to exact interpolation.

If we want to find another scenario where we would want to have this

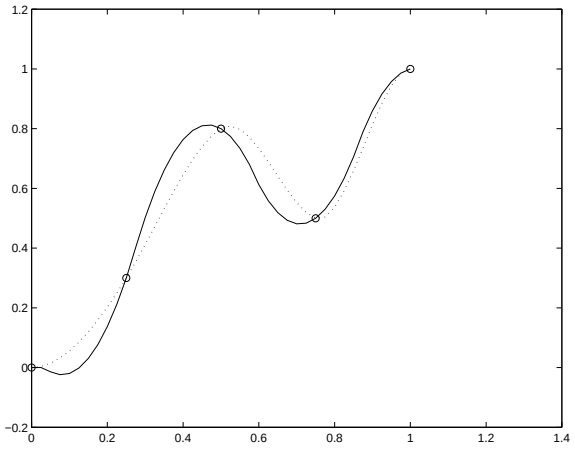


Figure 4: Planned curve through given points(solid) compared with the cubic spline curve (dotted). It can be worth noting that the maximal second derivative was 22.4 in the LP-case compared to 55.4 in the spline case, which must be considered as a significant improvement.

type of curves, apart from the desire to suppress noisy data, we could for instance easily imagine a situation where we are working with a robot, and where it is not crucial that the robot goes through the desired points exactly, but rather that we keep the second derivative as small as possible while keeping reasonably close to the interpolation points. In [13] this type of curve is generated with *smoothing splines*, that are trajectories generated by

minimizing

$$\sum_{k=1}^{n-1}(c^T x(t_k) - \gamma_k)^2 + \lambda \int_{t_0}^{t_N} u^2(t)dt. \tag{14}$$

Here, the γ_k:s are the interpolation points and the parameter $\lambda > 0$ controls the amount of smoothing. If it is small then the spline gets really close to the desired interpolation points, while a large λ makes the spline more smooth.

It is not hard to see how this idea can be used within our LP approach, and instead on minimizing (14), we chose as a cost function

$$\sum_{k=1}^{n-1}|c^T x(t_k) - \gamma_k| + \lambda z, \tag{15}$$

where λ is as before. The reason why we use $|\cdot|$ instead of $(\cdot)^2$ is of course due to the fact that we want the problem to be on LP form, using the same trick as before to transform a functional on the form $\min\{|w_1| + \ldots + |w_M|\}$ into a linear programming problem.

In principle, these extra terms in the cost function add nothing new to our concept since we just move the interpolation conditions from the constraints to the cost function in our LP formulation. Of course, we could have a function on the form

$$\sum_{k=1}^{n-1}\rho_k|c^T x(t_k) - \gamma_k| + \lambda z \tag{16}$$

instead of (15) if we want to penalize the distance to some interpolation

points more than others by varying the parameters ρ_k. This would not add anything substantially new to neither the formulation nor the solution of the problem

Instead of using the smoothing concept to get around the noisy data problem, we could, as described earlier, use intervals instead of points that we interpolate through.

However, if we only use the interval interpolation conditions, it is clear that we tend to move the curve close to one of the two end points on each interval, so if we combine this interval approach with the extra term in the cost function used in (15), we would get a trajectory that moves into the interior of the intervals instead. This is a property that could be desirable in some situations, and the result from these different cases can be seen in Figure 5-6.

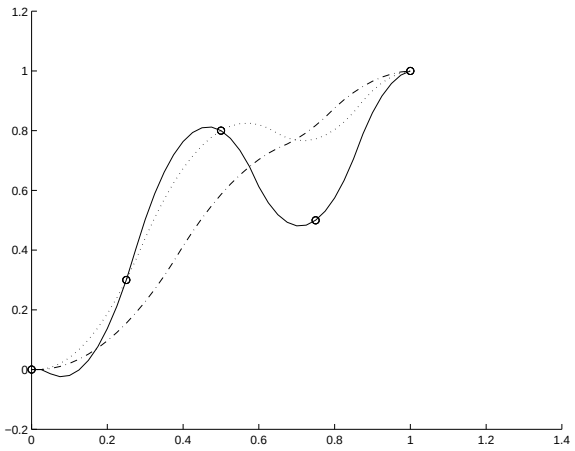


Figure 5: Smoothed trajectories with smoothing parameters $\lambda_1 = 0.02$ (solid), $\lambda_2 = 0.05$ (dotted) and $\lambda_3 = 0.1$ (dashed-dotted), which gave as maximal second derivative 22.3, 10.7 and 5.5 respectively. When λ gets high, we get more and more smoothness in the trajectory at the same time as less consideration is given to the interpolation points (circles).

4 Time Window Interpolation

4.1 A Bilinear Control System, Evolving in Virtual Time

The main idea now is to change our original system (1) into a system where we can control time explicitly. This is a desired feature of the system because

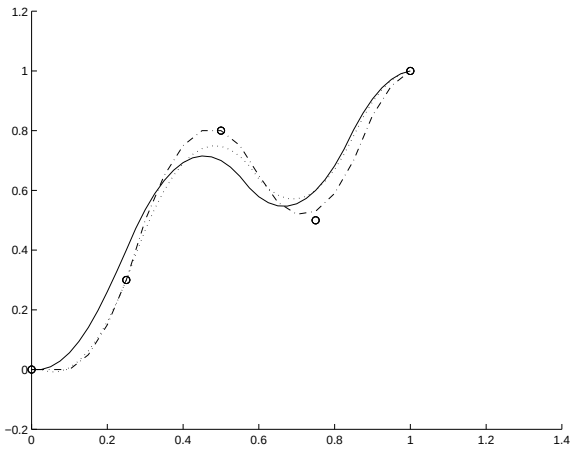


Figure 6: Trajectories that have to go through a given interval around each interpolation point (solid) which gave $u_{\max} = 15.1$, and the same conditions combined with an extra term in the cost function (dotted) to move the trajectory into the interior of the intervals. This gave $u_{\max} = 16.5$, with $\lambda = 0.04$. The last curve (dashed-dotted) corresponds to different weights on the different interpolation distances. Here we used the interpolation weights $\rho_1 = 1$, $\rho_2 = 1.2$ and $\rho_3 = 1.8$, which gave us $u_{\max} = 19.8$.

our aim is that we should be able to interpolate through intervals (8) not at time t_i, but at a time $t \in [\tau_{\star i}, \tau^{\star}{}_i]$. One idea, that will prove to be fruitful, is to introduce a virtual time, τ, and then let the system dynamics be given with respect to this time instead of the real one, t. We can now define a

differentiation with respect to τ by using the following notation

$$\frac{d\xi(\tau)}{d\tau} = \overset{\circ}{\xi}(\tau). \tag{17}$$

Then our wish to control the real time can be formulated as a controlled differential equation

$$\overset{\circ}{t} = \nu, \tag{18}$$

with ν as our control signal. This directly gives us that the original, continuous system dynamics, now evolving in the new, virtual time, τ, is

$$\overset{\circ}{x} = Ax\nu + bu\nu, \tag{19}$$

The nice thing here is that we can control time explicitly in (18). The price we have to pay for this is that our original linear control system (1) has been transformed into a bilinear system, (19). We thus have to give up all the nice properties that we used in the preceding section when formulating and solving the infinity norm, optimal control problem. But, as we will see in the next section, we do not have to feel too discouraged since it should be possible to use the solution to the linear control problem as a good starting guess when addressing this nonlinear, non-convex one.

The next thing we have to do is to see how this new system (19), evolving in the virtual time, τ, can be used in the time window optimal control

problem. If we discretize the virtual time axis this time instead of, as before, the real one, the cost function in the optimization problem will still be the same as before in (5). The reason why we don't need to change this part is of course due to the fact that we still want to solve the control problem for the original system (1), using the new one (19) as a mere tool for formulating the time window interpolation requirement in a systematic way. From (18) we directly see that the controlled discrete time equation becomes

$$\begin{aligned} & t_{i+1} = t_i + h_i \nu_i, \quad i = 1, \ldots, N-1 \\ & \nu_i \geq 0, \quad i = 1, \ldots, N \\ & t_1 = t_0, \; t_N = T, \end{aligned} \tag{20}$$

where ν_i are our new control signals. We need the extra constraint $v_i \geq 0$ because we do not want to allow for the possibility of having time starting to go backwards. If we now include the time window interpolation constrains, we get

$$\begin{aligned} & \tau_{\star j} \leq t_j \leq \tau^\star j, \quad j \in \mathcal{M}_\tau \\ & \alpha_j \leq c^T x_j \leq \beta_j, \quad j \in \mathcal{M}_\tau, \end{aligned} \tag{21}$$

where $\mathcal{M}_\tau$ is defined as $\mathcal{M}$ in the linear case, with the only difference that we now discretize with respect to τ instead of t. It can be worth pointing

out that if we wanted to, we could use other functions, $\phi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$, for formulating a generalized version of our interpolation requirement (21)

$$\phi(t_j, x_j) \leq 0, \tag{22}$$

and if, for instance, ϕ is on the form

$$\phi(t_j, x_j) = (t_j - \zeta_j)^2 + (c^T x_j - \xi_j)^2 - \delta_j^2, \tag{23}$$

we would be interpolating through circles around ξ_j with radius δ_j instead of rectangles. Here, however, we stick to (21) for the sake of computational efficiency since we then keep the time window constraints linear. We then get the total system on the form

$$\min \quad z$$

$$\text{when} \quad \begin{cases} u_i \le z \quad i = 1, \ldots, N-1 \\ -u_i \le z \quad i = 1, \ldots, N-1 \\ -\nu_i \le 0 \quad i = 1, \ldots, N-1 \\ t_{i+1} = t_i + h_i \nu_i \quad i = 1, \ldots, N-1 \\ x_{i+1} = x_i + h_i A x_i \nu_i + h_i b u_i \nu_i \\ \quad i = 1, \ldots, N-1 \\ x_1 = x_0, \quad x_N = X_T \\ t_1 = t_0, \quad t_N = T \\ \alpha_j - c^T x_j \le 0 \quad j \in \mathcal{M}_\tau \\ c^T x_j - \beta_j \le 0 \quad j \in \mathcal{M}_\tau \\ \tau_{\star j} - t_j \le 0 \quad j \in \mathcal{M}_\tau \\ t_j - \tau^\star{}_j \le 0 \quad j \in \mathcal{M}_\tau \end{cases} \tag{24}$$

There is, however, one major disadvantage with this system, and that is that it is non-convex. Therefore we need to chose a smart numerical strategy for solving it, as will be seen in the next subsection.

4.2 Numerical Strategy

Our optimization problem (24) could be formulated as

$$\min \qquad f(\bar{x})$$

$$\text{when} \quad \begin{cases} h_i(\bar{x}) = 0 & i = 1, \ldots, r \\ g_i(\bar{x}) \leq 0 & i = 1, \ldots, s \end{cases} \tag{25}$$

where f and the g_i:s are linear functions, while the h_i:s are bilinear ones. $\bar{x}$ is simply all the variables involved in the optimization problem (24),

$$\begin{aligned} \bar{x} &= (z, x_1^T, \ldots, x_N^T, t_1, \ldots, t_N, u_1, \ldots, u_{N-1}, \\ &\quad \nu_1, \ldots, \nu_{N-1})^T. \end{aligned} \tag{26}$$

The first order necessary optimality condition [8] is that

$$\nabla f(\bar{x}^\star) + \nabla h(\bar{x}^\star)\lambda + \nabla g(\bar{x}^\star)\mu = 0 \tag{27}$$

$$\mu^T g(\bar{x}^\star) = 0, \tag{28}$$

where $\nabla h(\bar{x})\lambda \stackrel{\triangle}{=} \sum_{i=1}^{r} \nabla h_i(\bar{x})\lambda_i$, (and similar for $\nabla g(\bar{x})\mu$), with feasibility conditions

$$h(\bar{x}^\star) = 0 \tag{29}$$

$$g(\bar{x}^\star) \leq 0, \tag{30}$$

where the Lagrange multipliers, λ and μ are in $\mathbb{R}^r$ and $\mathbb{R}^s$ respectively. If we now linearize the system at $\bar{x}_{k+1} = x_k + \Delta\bar{x}_k$, $\lambda_{k+1} = \lambda_k + \Delta\lambda_k$ and

$\mu_{k+1} = \mu_k + \Delta\mu_k$, (27)-(29) gives us a system of linear equations

$$\begin{pmatrix} L(\bar{x}_k, \lambda_k) & \nabla h(\bar{x}_k) & \nabla g(\bar{x}_k) \\ \nabla h(\bar{x}_k)^T & 0 & 0 \\ \mu_k^T \nabla g(\bar{x}_k)^T & 0 & g(\bar{x}_k)^T \end{pmatrix} \begin{pmatrix} \Delta\bar{x}_k \\ \Delta\lambda_k \\ \Delta\mu_k \end{pmatrix} =$$

$$= \begin{pmatrix} -\nabla f(\bar{x}_k) - \nabla h(\bar{x}_k)\lambda_k - \nabla g(\bar{x}_k)\mu_k \\ -h(\bar{x}_k) \\ -\mu_k^T g(\bar{x}_k) \end{pmatrix}, \tag{31}$$

where

$$L(\bar{x}_k, \lambda_k) \stackrel{\triangle}{=} \sum_{i=1}^{r} H_i(\bar{x}_k)\lambda_{ki}, \quad H_i(\bar{x}_k) = [\frac{\partial^2 h_i(\bar{x}_k)}{\partial \bar{x}_{kj} \partial \bar{x}_{kl}}]_{jl}. \tag{32}$$

The Hessians of f and the g_i:s vanish, since these are linear functions.

So, given $(\bar{x}_k, \lambda_k, \mu_k)$, we can calculate a descent direction $(\Delta\bar{x}_k, \Delta\lambda_k, \Delta\mu_k)$, using (31). We are not, however, going to use this direction in the way that was indicated above, but rather as a way to perform a line search in that direction. We then get

$$\begin{aligned} \bar{x}_{k+1} &= \bar{x}_k + \alpha_k \Delta\bar{x}_k \\ \lambda_{k+1} &= \lambda_k + \alpha_k \Delta\lambda_k \\ \mu_{k+1} &= \mu_k + \alpha_k \Delta\mu_k, \end{aligned} \tag{33}$$

where $\alpha_k \in [0, \alpha_{\max}]$ is the α that makes $(\bar{x}_{k+1}, \lambda_{k+1}, \mu_{k+1})$ minimize the

merit function

$$\begin{aligned} m(\bar{x},\lambda,\mu) &= \left|\nabla f(\bar{x}) + \nabla h(\bar{x})\lambda + \nabla g(\bar{x})\mu\right|^2 \\ &+ \left|h(\bar{x})\right|^2 + \left|\mu^T g(\bar{x})\right|^2, \end{aligned} \tag{34}$$

which is defined for measuring the progress of the iterative algorithm. $\alpha_{\max}$ is chosen as the maximal α for which we still have feasibility with respect to the inequality constraint (30)

$$\alpha_{\max} = \max\{\alpha \in [0,1] : g(\bar{x}_k + \alpha\Delta\bar{x}_k) \leq 0\} \tag{35}$$

We now have an iterative process for finding the point $x^\star$ that fulfills the first order necessary conditions for optimality.

The reason why we feel that this is all we need is that if we start with the solution to our fixed time LP-problem, which corresponds to setting $\nu_i \equiv 1$, $i = 1, \ldots, N-1$, and have small time windows, our approach should converge to, at least, a local optima. The results from this proposed method can be seen in Figures 7-8

5 Conclusions

We have developed a method for constructing trajectories that minimizes the infinity norm of the control imposed on a given linear system, while

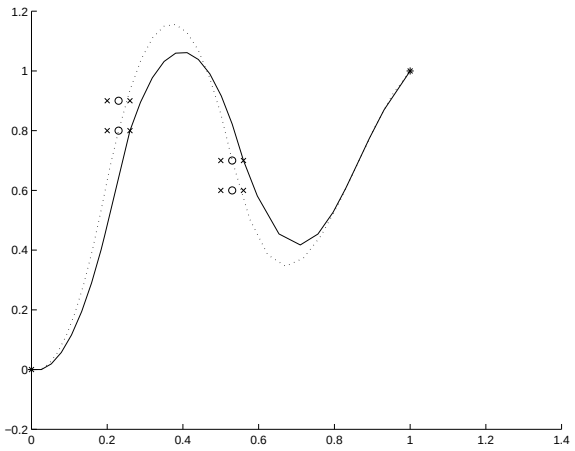
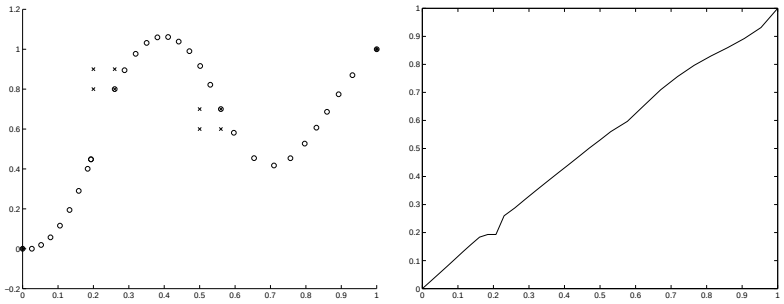


Figure 7: Fixed time interpolation curve (dotted) from Figure, with $u_{\max} = 36.1$, compared to the time window interpolation case (solid), interpolating through the same intervals as before, but through the time windows $[0.2, 0.26]$ and $[0.5, 0.56]$, where the dotted curve was used as a start guess in the iterative algorithm. This gave us a $u_{\max} = 27.7$, which must be considered as a significant improvement. In the figure, the x:s correspond to the corners of the windows that we are interpolating through.

the system generates outputs that moves between predefined points. This is a useful property, since in some applications it is important to keep the maximal value of the control input low instead of minimizing the average value or the produced energy. Even though our solution of the problem is not on a closed form, it is formulated as the optimum of a linear programming



(a) Distribution of the real time points

(b) Time as a function of the virtual time

Figure 8: The left figure shows how the real time points were distributed, while using an equidistant discretization of the virtual time between each of the four different virtual interpolation times. In the right figure, the real time, t, is plotted as a function of τ, and, as can be seen, the control v is close to one at all times.

problem. This is of importance since we then get access to all the efficient and well understood methods and theories dealing with linear programming.

We also generalize our method so that it can deal with situations where we do not want to interpolate exactly through the specified points, but rather focus on keeping the control signal small. This is useful, for instance, in statistical applications where noisy data makes it less important to interpolate

exactly through the desired points. For this, our method either uses an extra term in the cost function, penalizing the distance to the interpolation points, or interpolates through intervals instead of points. It also turns out that a combination of these two approaches could be used to move the trajectory into the interior of the interpolation intervals.

Two other, slightly different questions were investigated here as well, and the first one concerned how to formulate a time window interpolation requirement for linear control systems in a way that could be dealt with systematically. Our proposed solution was to introduce a virtual time and then let the real time be given through a controlled differential equation, evolving in the new, virtual time. This gave rise to a transformation of the original linear control system, so that it became a bilinear one that also evolved in the new, virtual time. From this transformation we gained a possibility of formulating the time window interpolation conditions, and we also saw that it was possible to formulate arbitrary time-state areas that we could interpolate through. However, the problem was that our new optimization problem now was a non-convex one.

The last problem was thus to come up with some sort of solution to this non-convex optimization problem. We proposed an iterative descent method

that used the solution to the original, fixed time interpolation problem as a start guess, and even though we couldn't prove global optimality, it was obvious from the example that we gained a lot when it came to reducing the value of the cost function. So, even though we do not know if we have found the global optima, we sure have found a way to find better solutions to the interpolation problem by allowing for a time window interpolation instead of just a fixed time interpolation requirement.

Acknowledgment

The authors would like to thank Andreas Nõu for helpful discussions and comments.

References

[1] O. Dahl and L. Nielsen: Torque-Limited Path Following by On-Line Trajectory Scaling, *IEEE Transactions on Robotics and Automation*, vol 6, No 5, Oct. 1990.

[2] M. Egerstedt, X. Hu, H. Rehbinder and A. Stotsky: Path Planning and Robust Tracking for a Car-Like Robot, Proceedings of the *5th Sympo-*

sium on Intelligent Robotic Systems, Stockholm, Sweden, 1997.

[3] P. Enqvist, C.F. Martin, J. Tomlinson and Z. Zhang: Linear Control Theory, Splines and Interpolation, *Computation and Control IV,* Birkhauser, 1995.

[4] L. Faybusovich and J.B. Moore: Infinite Dimensional Quadratic Optimization: Interior-Point Methods and Control Applications, to appear in *Journal of Applied Mathematics and Optimization.*

[5] J-C. Latombe: *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[6] G. Leitmann: *The Calculus of Variations and Optimal Control*, Plenum Press, New York, 1981.

[7] D.G. Luenberger: *Optimization by Vector Space Methods*, John Wiley and Sons, Inc., New York, 1969.

[8] D.G. Luenberger: *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, 1989.

[9] Z. Luo and G. Wahba: Hybrid Adaptive Splines, *Journal of the American Statistical Association*, vol 92, N437, 1997.

[10] C.F. Martin, J. Tomlinson and Z. Zhang: Splines and Linear Control Theory, to appear in *Acta Applicandae Mathematicae.*

[11] G.J. Papas: Avoiding Saturation by Trajectory Reparametrization, *Proc. of the 35th Conference on Decision and Control*, Kobe, Japan, Dec. 1996.

[12] G.Wahba: *Spline Models for Observational Data*, Society for Industrial and Applied Mathematics, 1990.

[13] E.J. Wegman and I.W. Wright: Splines in Statistics, *Journal of the American Statistical Association*, vol 78, N382, 1983.