Trajectory Generation Using Sum-of-Norms Regularization

Henrik Ohlsson, Fredrik Gustafsson, Lennart Ljung and Stephen Boyd

Abstract—Many tracking problems are split into two subproblems, first a smooth reference trajectory is generated that meet the control design objectives, and then a closed loop control system is designed to follow this reference trajectory as well as possible. Applications of this kind include (autonomous) vehicle navigation systems and robotics. Typically, a spline model is used for trajectory generation and another physical and dynamical model is used for the control design. Here we propose a direct approach where the dynamical model is used to generate a control signal that takes the state trajectory through the waypoints specified in the design goals. The strength of the proposed formulation is the methodology to obtain a control signal with compact representation and that changes only when needed, something often wanted in tracking. The formulation takes the shape of a constrained least-squares problem with sum-of-norms regularization, a generalization of the $\ell_1$-regularization. The formulation also gives a tool to, e.g. in model predictive control, prevent chatter in the input signal, and also select the most suitable instances for applying the control inputs.

I. INTRODUCTION

Consider a dynamic system with output $y(t)$ where the design objectives can be formulated as

$$y(t_k) \approx W(t_k), \quad t_k \in T. \tag{1}$$

These points, $W(t_k)$, will be referred to as waypoints. The conventional approach is based on the following two steps:

1. Generate spline functions between the waypoints to get a smooth trajectory $y_{ref}(t)$.
2. Design a control system that generates a control input $u(t)$ that makes the system output as close as possible to $y_{ref}(t)$.

A potential problem is that the reference trajectory $y_{ref}(t)$ may not be feasible for the system in that it varies too fast to allow the dynamics to follow the trajectory with given input saturations. Conversely, it may vary too slowly, giving a conservative control performance.

The suggested approach circumvents this problem by generating the reference trajectory based on the system dynamics, rather than spline functions. The result is an input sequence \{u(t)\}, with a compact representation, that gives the output sequence \{y(t)\} that passes (within a given distance to) the waypoints. In the presence of disturbances and process noise, a feedback control is still necessary, but this provide small corrections to the already computed input sequence.

The advantages of the method include:

- The complexity is linear in time.
- Control signal saturations can be incorporated.
- Different constraints on input smoothness can be incorporated, as piecewise linear or constant inputs. This is relevant in applications for which a change in control signal is associated with a cost (maybe communicating a change is costly) or the storage is limited.
- State constraints can be added for forbidden regions in the state space.
- The trajectory can be forced to pass the waypoints or any specified distance to them.

Example 1 (Two Dimensional Tracking Problem):

Considering a two dimensional tracking problem. Assume that we would like the system output to take the values

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 \\ -10 \end{bmatrix}, \begin{bmatrix} 20 \\ 0 \end{bmatrix}, \begin{bmatrix} 30 \\ 0 \end{bmatrix}, \begin{bmatrix} 30 \\ 10 \end{bmatrix}, \begin{bmatrix} 20 \\ 10 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2}$$

at

$$t = 0, 1, 2, 3, 4, 4.5, 5, 6. \tag{3}$$

(2) and (3) here define our waypoints. Assume that the control signal $\|u\|^2 < 40$. We may also assume that its impractical to communicate or to store an output reference as a look-up table with a value for each sample time. This is often the case for industrial robots and autonomous vehicle navigation systems. The solution is to fit a spline to the waypoints and use this as a reference trajectory. In this particular example, the spline would have 8 breakpoints (one for each waypoint). A feedback controller is then applied to follow the spline reference trajectory. Since the spline fitting problem is commonly seen only as a geometrical task and no consideration to the dynamical system and input constraint are taken, the spline reference may be impossible to follow. We will come back to this problem later in Section IV.

II. PROBLEM FORMULATION

Given a system

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4}$$

$$y(t) = Cx(t)$$

$x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and a set of so called waypoints

$$W = \{W(t), t \in T\}, \quad T = \{t_k, k = 1, \ldots, M\}. \tag{5}$$

The problem is to find $u$ so that the output of (4) closely follows the waypoints:

$$y(t_k) \approx W(t_k), \quad t_k \in T \tag{6}$$
This should be solved under a number of constraints:

- Given a time grid $T_{\text{grid}}$, based on the sampling time $T_s$: $T_{\text{grid}} = \{t_s = sT_s, s = 1, \ldots, N\}$. Assume that $T$ is a subset of $T_{\text{grid}}$.
- The $p$th derivative of $u$ is an impulse train on the grid $T_{\text{grid}}$:
  $$u^{(p)}(t) = \sum_{k=1}^{N} v_k \delta(t - kT_s), \quad v_k \in R_{nu}$$ (7)
- As many as possible of the terms $v_k$ in (7) should be zero.
- There are input and state constraints (on the grid $T_{\text{grid}}$):
  $$x(kT_s) \in K$$
  $$u(kT_s) \in U$$ (8)

We should comment on (7). The common way to impose a smooth output, which often is desirable (see e.g. [5], [14]), is by using a smooth spline as a reference. But if (7) we impose a smoothness constrain on the control input and also implicitly on the output.

That many of the $v_k$s are zero imply a compact representation of the input signal. This may be advantageous when storage is limited. It also means few changes in the control signal. This may save money by reduced communication but may also save actuators (see discussions on chattering in model predictive control (MPC), see e.g. [16] or [10]).

### III. Proposed Method

Let the upper part of the extended vector $X(t)$ be the state $x(t)$ of (4), while the lower part is made up of the $p - 1$ derivatives of $u$,

$$X(t) = [x(t) \quad u(t) \quad \dot{u}(t) \quad \ldots \quad u^{(p-1)}(t)]^T.$$ (9)

Let us accordingly extend the model (4) by ($I_r$ is the $r \times r$ unit matrix):

$$\dot{X}(t) = \bar{A}X(t) + \bar{B}u^{(p)}(t)$$
$$y(t) = \bar{C}X(t)$$ (10a)

$$\bar{A} = \begin{bmatrix} A & B & 0 & 0 & \ldots & 0 \\ 0 & 0 & I_m & 0 & \ldots & 0 \\ 0 & 0 & 0 & I_m & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix}$$ (10c)

$$\bar{B} = \begin{bmatrix} 0 & \ldots & 0 & I_m \end{bmatrix}^T$$ (10d)

$$\bar{C} = [C \quad 0 \quad \ldots \quad 0]$$ (10e)

Then by sampling this system with a pulse-train as input (**c2d(syst, Ts, ′imp′)** in Matlab) we obtain

$$X(kT_s + T_s) = FX(kT_s) + Gv_k$$ (11a)
$$x(kT_s) = PX(kT_s)$$ (11b)
$$y(kT_s) = HX(kT_s)$$ (11c)
$$u(kT_s) = RX(kT_s)$$ (11d)

with

$$F = e^{\bar{A}T_s}$$
$$G = F\bar{B}$$
$$H = \bar{C}$$
$$P = \begin{bmatrix} I_n & 0 & 0 & \ldots & 0 \\ 0 & I_m & 0 & \ldots & 0 \end{bmatrix}$$ (12d)
$$R = \begin{bmatrix} 0 & I_m & 0 & \ldots & 0 \end{bmatrix}$$ (12e)

Let us assume that $x(0)$ is known and $u(0) = \dot{u}(0) = \ldots = u^{(p-1)}(0) = 0$ for simplicity. We can now phrase the problem as

$$\min_{v_k \in U} \sum_{t \in T} \|y(t) - W(t)\|^2$$ (13)

w.r.t. $v_k$ under the constraints (11a), (8) and trying to have as many $v_k$ as possible equal to zero. To capture the latter we wish to use sum-of-norms regularization:

$$\min_{v_k \in U} \sum_{t \in T} \|y(t) - W(t)\|^2 + \lambda \sum_{k=1}^{N} \|v_k\|_p$$ (14a)

$$u(kT_s) \in U$$ (14b)

$$x(kT_s) \in X$$ (14c)

where $y(kT_s)$, $x(kT_s)$ and $u(kT_s)$ are generated from $v_k$ and $X(0)$. $\|\cdot\|_p$ is defined as $\|z\|_p \triangleq \left(\sum_{i=1}^{n} |z_i|^p\right)^{1/p}$ for a vector $z = [z_1 \ z_2 \ldots \ z_n]^T$.

The last term in the cost is a regularization term, and $\lambda$ is a positive constant that is used to control the trade-off between the fit to the waypoints $W(t)$ (the first term) and the size of the state changes caused by $v_k$ (the second term).

The regularization norm could actually be any $\ell_p$-norm, like $\ell_1$ or $\ell_2$, but it is crucial that it is a sum of norms, and not a sum of squared norms.

When the regularization norm is taken to be the $\ell_1$ norm, i.e., $\|z\|_1 = \sum_{k=1}^{N} |z_k|$, the regularization in (14a) is a standard $\ell_1$ regularization of the least-squares criterion. Such regularization has been very popular recently, e.g. in the much used Lasso method, [15] or compressed sensing [2], [1]. See also [7] and [11] for relevant contributions.

There are two key reasons why the criterion (14a) is attractive:

- It is a convex optimization problem, so the global solution can be computed efficiently. In fact, its special structure allows it to be solved in $O(N)$ operations, so it is quite practical to solve it for a range of values of $\lambda$, even for large values of $N$.
- The sum-of-norms form of the regularization favors solutions where “many” (depending on $\lambda$) of the regularized variables come out as exactly zero in the solution. In this case, this implies that many of the estimates of $v_k$ become zero (with the number of $v_k$s becoming zero controlled roughly by $\lambda$).

A third advantage is that

- It is easy to include realistic state constraints without destroying convexity.
We should comment on the difference between using an $\ell_1$ regularization and some other type of sum-of-norms regularization, such as sum-of-Euclidean norms. With $\ell_1$ regularization, we obtain an estimate of $v_k$ having many of its elements equal to zero. When we use sum-of-norms regularization, the whole estimated vector $v_k$ often becomes zero; but when it is nonzero, typically all its elements are nonzero. In a statistical linear regression framework, sum-of-norms regularization is called Group-Lasso [17], since it results in estimates in which many groups of variables are zero.

One final step is also useful. From our estimate of $v_k$ from (14a), we simply carry out a final least-squares fit over the nonzero $v_k$ (fixing the other $v_k$ to zero). This typically gives a small improvement in fit to the waypoints.

A. Solution Algorithms and Software

Many standard methods of convex optimization can be used to solve the problem (14a). Systems such as CVX [4], [3] or YALMIP [9] can readily handle the sum-of-norms regularization, by converting the problem to a cone problem and calling a standard interior-point method. For the special case when the $\ell_1$ norm is used as the regularization norm, more efficient special purpose algorithms and software can be used, such as 1L_1.als [8]. Recently many authors have developed fast first order methods for solving $\ell_1$ regularized problems, and these methods can be extended to handle the sum-of-norms regularization used here; see, for example, [12, §2.2]. Both interior-point and first-order methods have a complexity that scales linearly with $N$.

IV. NUMERICAL ILLUSTRATION

Example 2 (Two Dimensional Tracking Problem, Cont’d):
Let us return to Example 1. Assume that we chose to model our system with the model (see Chapter 13 of [6], there called a constant acceleration model)

$$\dot{x}(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ I_2 \end{bmatrix} u(t),$$

(15)

$$y(t) = \begin{bmatrix} I_2 \\ 0 \\ 0 \end{bmatrix} x(t).$$

The two first elements of the state $x$ are the $x$- and $y$-position, the third and fourth, velocity in the $x$- and $y$-direction and the two last elements, the acceleration in $x$- and $y$-direction. $u$ is the jerk (the derivative of the acceleration). Assume now that we believe that a piecewise constant input $u$ in (15), (that means that $p = 1$ in (7)) gives a smooth enough output. The requirement of a piecewise constant input implies that we have to extend our model with a integrator. We obtain the extended model

$$\dot{x}(t) = \begin{bmatrix} 0 & I_2 & 0 & 0 \\ 0 & 0 & I_2 & 0 \\ 0 & 0 & 0 & I_2 \\ 0 & 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \sum_k v_k \delta(t - kT_s) \end{bmatrix},$$

(16)

and seek a “sparse” pulse train $\{v_k\}$. Discretizing (16) under the assumption that $v_k$ is a pulse train and with $T_s = 0.1$ gives (see (11) and (12))

$$X(kT_s + T_s) = \begin{bmatrix} I_2 & 0.1I_2 & 0.005I_2 & 0.0002I_2 \\ 0 & I_2 & 0.1I_2 & 0.005I_2 \\ 0 & 0 & I_2 & 0.1I_2 \\ 0 & 0 & 0 & I_2 \end{bmatrix} X(kT_s) + \begin{bmatrix} 0.0002I_2 \\ 0.005I_2 \\ 0.1I_2 \\ I_2 \end{bmatrix} v_k,$$

$$y(kT_s) = \begin{bmatrix} I_2 \\ 0 \\ 0 \end{bmatrix} X(kT_s).$$

(17)

Let $X(0) = 0$ and use (14a) (with (17) as a constraint) to compute $\{v_k\}$. Finally carry out a least-squares fit over the nonzero $v_k$ (fixing the other $v_k$ to zero). The trajectory generated by this last estimate $\{v_k\}$ is shown in Figure 1 for $\lambda = 0.05, 0.1$ and 0.5 and $\ell_1$-norm. The associated pulse train is given in Figure 2. Feeding a system consisting of a single integrator gives the searched piecewise constant input signal that should be used in (15) to make the output as in Figure 1. Notice that it is only 10, 9 respective 6 $v_k$-values that are needed to represent the control input. A known control saturation would also be easy to implement and impossible reference trajectories (as may occur when using splines) are not a problem.

Fig. 1. The computed trajectory ($x$- and $y$-position) of Example 2 shown for $\lambda = 0.05, 0.1$ and 0.5 (black, gray resp. light gray line). Waypoints are shown with filled circles.

Let us also compute the trajectory using a pulse train, piecewise constant and piecewise linear input ($p = 0, 0 = 1$ resp. $p = 2$ in (7)). The result using $\lambda = 0.05$ is shown in Figures 3 and 4. Note that the black line in Figure 1 thus coincides with the gray line in Figure 3.
**Example 3 (Selecting Time Instances for Control Inputs):**

Consider the DC-motor model

\[
\begin{align*}
\dot{x}(t) &= \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t) \\
y(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x(t).
\end{align*}
\]

Let \( W \) contain the reference values

\[ \{0, 0, 0, 0, 10, 0, 0, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10\} \]

and let the associated \( t_k \) be

\[ \{0.75, 2.25, 3, 3.75, 5.25, 7.5, 7.8, 8.25, 9, 10.5, 12, 13.5, 15\}. \]

Extend now (18) by adding an extra state to be able to impose a piecewise constant input \( u(t) \). Set the sampling time \( T_s = 0.15 \) and discretize. The extended discretized model takes the form

\[
X(kT_s + T_s) = \begin{bmatrix} 0.8607 & 0 & 0.1393 \\ 0.1393 & 1 & 0.0107 \\ 0 & 0 & 1 \end{bmatrix} X(kT_s) + \begin{bmatrix} 0.1393 \\ 0.0107 \\ 1 \end{bmatrix} v_k
\]

\[
y(kT_s) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} X(kT_s).
\]

The computed \( y(t) \) using \( \lambda = 0.5, \|v_k\|_2 < 40 \) and an initial state \( x(0) = \begin{bmatrix} 0 & 2 \end{bmatrix}^T \) is shown in Figure 5. The associated pulse train is given in Figure 6.

Let us now make the example a bit more interesting by applying the technique repeatedly, optimizing over a horizon
of $mT_s$ and applying the control for $nT_s$ seconds before re-optimizing. Assume the same waypoints as above. With $m = 14$, $n = 4$ and $\lambda = 1$ the result shown in Figure 7 and 8 was obtained. To recursively compute control signals like this is done in optimal control and MPC. To avoid changing the control signal, if not necessary, like above, may help prevent chattering in MPC (see e.g. [16] or [10] for relevant contributions discussing the problem of chattering in applications).

---

**V. Conclusion**

A spline representation is often chosen for the reference signal in tracking application. The motivation for this is twofold:

- Using splines a certain degree of smoothness can be guaranteed.
- Splines can be compactly represented.

A spline representation has two disadvantages:

- A spline is a piecewise polynomial function with the different pieces often glued together at the waypoints (see for example [13] for an attempt to remove this constraint). A more flexible approach would be to not restrict the breakpoints of the spline to the waypoints. This could for example lead to a smoother reference trajectory with a more compact representation.
- A second disadvantage is that it is difficult to guarantee that the generated reference is physically possible for the system to follow.

The proposed method generates a control input which could be fed through the system model to give a spline. We see no reason for computing this spline, however. The output of the method is rather the sequence \( \{v_k\}_{k=1}^{N} \). The sparse sequence \( \{v_k\}_{k=1}^{N} \) can be stored or communicated using limited resources to the system. The system can then generate an input by integrating the pulse train defined by \( \{v_k\}_{k=1}^{N} \) which takes the system through the specified waypoints. A feedback control is still necessary to reduce the effect of noise and model errors.

The proposed method can hence guarantee a smooth system output. It does not have the problem of generating infeasible reference trajectories. And, since the output sequence is optimized to have few changes but at suitable instances, the representation may be considerably more compact then using splines as a reference.

The proposed method has an optimization formulation. It is convex and the complexity grows linear with \( N \). Constraints, such as control signal saturations, can easily be incorporated. Relations to optimal control and MPC have also been discussed. There is also a relation to Lebesgue sampling, even-triggered sampling and control (see e.g. [18]). This has not been discussed and is seen as future work.
VI. ACKNOWLEDGMENTS

Partially supported by the Swedish foundation for strategic research in the center MOVIII and by the Swedish Research Council in the Linnaeus center CADICS. The authors also want to thank Professor Bo Bernhardsson for useful comments on the manuscript.

REFERENCES


