Expanding Motion Programs Under Input Constraints

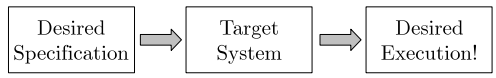
Patrick Martin and Magnus Egerstedt
pmartin@ece.gatech.edu, magnus@ece.gatech.edu
Georgia Institute of Technology, Atlanta, GA 30332, USA

***Abstract*— We explore the effect that bounded inputs have when we specifying sequences of control laws for driving a linear, dynamical system through intermediary equilibrium points. These sequences, given in the Motion Description Language (MDL) formalism, may not be feasible in that they may violate the bounded input constraints. To overcome this problem, we augment the MDL sequences by inserting extra control modes, resulting in a new motion program that does in fact satisfy the input constraints. We first consider the problem of "compiling" the motion program for feasibility, and then propose a method that constructs new motion programs based on the input bounds, system dynamics, and task design goals.**

I. Introduction

For complex systems, one design methodology that has proven useful is to decompose the control task into sequences of primitive building blocks, designed with a particular objective in mind. These building blocks are then sequenced together to produce the desired, overall system behavior. One way in which this sequencing of motion programs can be done is in the framework of Motion Description Languages (MDL), consisting of strings of controller-interrupt pairs, e.g. [2], [8], [9]. For example, in mobile robotics, one can construct motion programs with MDL that define a sequence of behaviors for navigation through an environment.

When specifying such motion programs, it is generally assumed that the controller "fits" the dynamics of the system in that the system can execute the control string. However, in a number of practical applications, e.g. embedded control systems, there are hard constraints on the actuator signals achievable that effect what motion programs can be executed. It is thus possible that a motion program that is intended to perform some action, actually fails to accomplish the task because of the input constraints. We can visualize this specification and execution mismatch via the illustration in Figure I.



(a) Typical MDL design process.

(b) Executed MDL program.

Fig. 1. Typically, one specifies a control program and assume (hope) that the execution will be as intended, as visualized in 1(a). More than likely, however, undesirable behaviors caused by the system capabilities and environment will be the result 1(b).

This problematic specification-to-execution process will cause more issues as we move into cyber-physical system (CPS) application domains, where the interactions with the physically constrained environment play a key role. As such, we can no longer assume that the motion programs will execute correctly on a cyber-physical platform. This paper proposes a method that opens up an area of research into "MDL compilers" for CPS. We lay the foundation for more exploration into how theory and algorithms can be developed to compile our MDL programs into executable code that satisfies the constraints of the CPS application.

In particular, this paper considers linear systems that must operate under bounded input constraints. Understanding how bounded input can affect the control and output regulation of linear systems has been considered as far back as [11]. Additionally, the recent work in [5] considered the H_2 synthesis problem where the input signal is bounded by some maximum value. However, This prior work did not address switching between controllers that must each satisfy the bounded input constraints.

This paper focuses on developing an algorithm to expand a motion program in order to prevent the input from violating the actuator constraints. There have been several recent developments for maintaining the input bounds for switched systems, e.g. [6], [10], [13]. The work in [6] devised a supervisory control algorithm that selects a controller from a set to achieve performance while balancing stability requirements under input and state constraints. The work in [10], [13] use a Lyapunov-based scheduling procedure to select gains for controllers to maintain system stability. Our method for compiling motion programs with bounded inputs differs from this prior work by focusing on minimizing the number of mode insertions in the motion program string, while maintaining the bounded input constraints.

The outline of this paper is as follows: In Section II, we discuss MDL and relate them to the bounded input problem under consideration in this paper. In Section III, we derive regions of attraction for the individual control modes, followed by the MDL compiler, in Section IV.

II. Background

In this section, we provide the background to MDL as well as discuss how to frame the effects of bounded inputs on such motion program specifications.

A. Motion Description Languages

The work of [2], [8] describe MDLs as languages for composing global motion programs from collections of predefined controllers. these controllers are paired with an interrupt functions that cause transitions between the controllers at execution. More specifically, let the system have dynamics of the form

$$\dot{x} = f(x,u),\ x \in \mathcal{X} \subseteq \mathbb{R}^n, u \in \mathcal{U} \tag{1}$$

where x is the state of the system, and u is the control input. If we let the control input be given by a closed-loop mapping $\kappa : \mathcal{X} \to \mathcal{U}$, and introduce interrupt functions as mappings $\xi : \mathcal{X} \to \{0,1\}$, where 1 indicates that an interrupt has occurred, we denote a MDL mode by the tuple (κ, ξ). The interpretation here is that the system in (1) executes the controller κ until $\xi \to 1$. A MDL program thus becomes a sequence of such controller-interrupt pairs.

In this paper, we will restrict the MDLs to take on a particularly simple form, in that we will insist on the control laws being affine in the state, driving the system to particular equilibrium points, and the interrupts triggering when these points are reached.

B. MDLs for Moving Between Set-Points

Consider the unstable, controllable linear system

$$\dot{x} = Ax + bu,\ x \in \mathbb{R}^n,\ u \in \mathbb{R}. \tag{2}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}$, and A, b are matrices of appropriate dimension. A stabilizing controller for this system can be generated by solving the LQ design problem where the cost functional is

$$J(x,u) = \int_0^\infty \left(x^T Q x + \sqrt{2} u^T u \right) dt.$$

The solution is given by the feedback gain P that solves the Riccati equation

$$A^T P + PA + Q - 2Pbb^T P = 0, \tag{3}$$

where $Q \succ 0$. The resulting feedback control for stabilizing (2) is thus given by

$$u = -b^T P x. \tag{4}$$

What we will do is select such a P and then augment the controller with an affine term that drives the system to a given set-point. By stringing together different such affine terms, we get a motion programs that takes the system through a sequence of set-points. Consequently, we modify the control input to include an affine term, v, as

$$u = -b^T P x + v. \tag{5}$$

By applying the controller (5) to (2), we get the following closed loop dynamics:

$$\dot{x} = (A - bb^T P)x + bv,$$

with globally attractive, stationary point given by

$$x_v = -(A - bb^T P)^{-1} bv.$$

(Note that the inverse is well defined since the real parts of the eigenvalues of $(A - bb^T P)$ are negative by design.) As such, we can defined the interrupt function $\xi(P, v, \epsilon)$ that triggers once the state of the system is close to x_v, i.e.

$$\xi(P,v,\epsilon) = \begin{cases} 1 & \text{if } \|x - x_v\|_P^2 \leq \epsilon \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where $\|z\|_P^2 = z^T P z$.

As such, we let the motion program used for driving the system through a collection of set-points be given by strings $((P, v_1), \xi(P, v_1, \epsilon_1)), \ldots, ((P, v_N), \xi(P, v_N, \epsilon_N))$. As we have assumed that P is fixed, we can use the shorthand $(v_1, \epsilon_1), \ldots, (v_N, \epsilon_N)$ to denote these MDL strings.

III. Compiling MDLs under Input Constraints

For each $v \in \mathbb{R}$, we thus have a controller that takes the system (asymptotically) to the set-point x_v. The first thing we need to do in order to "compile" the MDL programs is to characterize what the set of such set-points actually looks like under the input constraint $|u| < u_{max}$.

A. Regions of Attraction

Let the set of stationary points be denoted by $\mathcal{X}$. In order to calculate the regions of attraction around each point, we need the following lemma:

Lemma 1 (Stationary Points under Bounder Input): Let (A, b) be a controllable pair, let P be the positive definite matrix solution to the Riccati equation (3) for some $Q \succ 0$, and let $K = (A - bb^T P)$. If $u = -b^T P x + v$ and $|u| < u_{max}$, then the set of stationary points $\mathcal{X}$ is given by

$$\mathcal{X} = -K^{-1} b (b^T P K^{-1} b + 1)^{-1} [-u_{max}, u_{max}]$$

if $b^T P K^{-1} b + 1 \neq 0$ or,

$$\mathcal{X} = -K^{-1} b \mathbb{R},\ \text{otherwise.}$$

Proof: Assume that $x_{v_i} \in \mathcal{X}$ is the stationary point obtained by using the open-loop control v_i in equation (5). Then, the total control effort needed to hold the system at x_{v_i} is

$$\begin{aligned} u_{v_i} &= -b^T P x_{v_i} + v_i \\ &= (b^T P K^{-1} b + 1) v_i \end{aligned}$$

Note that if $b^T P K^{-1} b + 1 = 0$ then $u_{v_i} = 0$ for any $v \in \mathbb{R}$. If the equality *does not* hold, then the choice of v must come from the set

$$v \in \mathcal{V} = (b^T P K^{-1} b + 1)^{-1} [-u_{max}, u_{max}].$$

in order to maintain that $|u| < u_{max}$; otherwise, $v \in \mathbb{R}$. Hence, the set of stationary points is

$$\mathcal{X} = -K^{-1} b \mathcal{V}$$

which completes the proof. ∎

Now, with a proper characterization of these stationary points, we can proceed with calculating the regions of the state space from which these points can be reached

with bounded inputs. These regions will form the basis for developments in the following sections.

Theorem 1 (Regions of Attraction Under Bounded Input): Given the assumptions in Lemma 1, the region of attraction around the point x_{v_i} is given by

$$\mathcal{E}(P, v_i) = \{x \in \mathbb{R}^n | (x - x_{v_i})^T P (x - x_{v_i}) \leqslant \alpha_{v_i}\} \quad (7)$$

with

$$x_{v_i} = -K^T b v_i$$

and

$$\alpha_{v_i} = (b^T P b)^{-1} (u_{max} - |u_{v_i}|)^2. \quad (8)$$

Proof: Let $x = x_{v_i} + \Delta x_{v_i}$, then

$$u = -b^T P (x_{v_i} + \Delta x_{v_i}) = u_{v_i} - b^T P \Delta x_{v_i}.$$

However, since we have the constraint $u \in [-u_{max}, u_{max}]$, we interpret the above equation as

$$b^T P \Delta x_{v_i} \in [-u_{max} + u_{v_i}, u_{max} + u_{v_i}].$$

P is a solution to the Riccati equation (3), so we define the function

$$V(\Delta x_{v_i}) = \Delta x_{v_i}^T P \Delta x_{v_i},\ \forall \Delta x_{v_i} \in \mathbb{R}^n,\ \Delta x_{v_i} \neq 0. \quad (9)$$

If this function is Lyapunov, then it can serve as a conservative region of attraction around the point x_{v_i}. To show this fact, consider the ellipsoid generated by $\Delta x_{v_i}^T P \Delta x_{v_i} = \gamma$, where $\gamma > 0$ and real and $\forall \Delta x_{v_i} \in \mathbb{R}^n$. Assume γ is chosen such that $|-b^T P \Delta x_{v_i}| < u_{max}$, and, furthermore, we choose some $\beta \in (0, \gamma)$.

Thus, we want to solve the following maximization problem for any $\Delta x_{v_i} \in \mathbb{R}^n$:

$$\begin{aligned} &\max_{\Delta x_{v_i}} b^T P \Delta x_{v_i} \\ &\text{s.t. } \Delta x_{v_i}^T P \Delta x_{v_i} = \gamma. \end{aligned}$$

Forming the Lagrangian,

$$L = b^T P \Delta x_{v_i} - \lambda (\Delta x_{v_i}^T P \Delta x_{v_i} - \gamma),$$

and setting its derivative w.r.t. Δx_{v_i} equal to 0 we get that

$$\Delta x_{v_i} = -\frac{1}{2\lambda} b \quad (10)$$

which implies that the maximum Δx_{v_i} is parallel to the input matrix b. According to the constraint equation we calculate the value of λ and insert into equation (10), resulting in the maximum value:

$$\Delta x_{max} = \sqrt{\frac{\gamma}{b^T P b}} b.$$

Using this value in the control law results in the maximum control effort

$$u_\gamma = b^T P \sqrt{\frac{\gamma}{b^T P b}} b = \sqrt{\gamma} \|b\|_P$$

Finally, if we compare the difference between the maximum input along the the γ level-set and that on the β level-set we get

$$u_\beta - u_\gamma = (\sqrt{\beta} - \sqrt{\gamma}) \|b\|_P$$

which is a strictly negative quantity, by the assumption that $\beta \in (0, \gamma)$. Therefore, the control effort reduces as the state decays within the ellipsoid $\Delta x_{v_i}^T P \Delta x_{v_i}$, and (9) is a Lyapunov equation.

Now that we know the region defined by (9) is Lyapunov, we find the solution to

$$\min_{\Delta x_{v_i}} \Delta x_{v_i}^T P x_{v_i}$$

such that

$$b^T P \Delta x_{v_i} = -u_{max} + u_{v_i}$$

or

$$b^T P \Delta x_{v_i} = u_{max} + u_{v_i}.$$

Letting $c = Pb$ and $d_\pm = \pm u_{max} + u_{v_i}$, the Lagrange necessary and sufficient conditions (see for example [7]) for these quadratic optimization problems are:

$$\begin{aligned} P \Delta x_{v_i} + c \lambda_{v_i} &= 0 \\ c^T \Delta x_{v_i} - d_\pm &= 0 \end{aligned}$$

where $\lambda_{v_i} \in \mathbb{R}^n$ is the Lagrange multiplier. Solving these equations results in

$$\begin{aligned} \lambda_{v_i} &= -(c^T P^{-1} c)^{-1} d_\pm \\ \Delta x_{v_i} &= P^{-1} c (c^T P^{-1} c)^{-1} d_\pm. \end{aligned}$$

Inserting the solution for Δx_{v_i} into (9) results in:

$$\Delta x_{v_i}^T P \Delta x_{v_i} = (b^T P b)^{-1} (\pm u_{max} + u_{v_i})^2$$

where $(b^T P b)^{-1}$ exists since $P \succ 0$ and $b \neq 0$ by our controllability assumption. We choose the smallest and define it as

$$\alpha_{v_i} = (b^T P b)^{-1} (u_{max} - |u_{v_i}|)^2.$$

Therefore, the region around each stationary point x_{v_i} where $|u| \leq u_{max}$ is given by

$$\mathcal{E}(P, v) = \{x \in \mathbb{R}^n | (x - x_{v_i})^T P (x - x_{v_i}) \leq \alpha_{v_i}\},$$

as in equation (7). ■

A corollary of Theorem 1 describes the characterization of the entire region of attraction about the points in $\mathcal{X}$:

Corollary 1: The region of attraction around the set of stationary points $\mathcal{X}$ is given by

$$\mathcal{L} = \bigcup_{v_i \in \mathcal{V}} \mathcal{E}(P, v_i).$$

Figure 2 shows an example of the application of Theorem 1 to (2) with matrices

$$A = \begin{pmatrix} 1 & 2 \\ 0 & -1 \end{pmatrix},\ b = \begin{pmatrix} 2 \\ 1 \end{pmatrix},\ Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and $u_{max} = 1$. This figure shows some of the stationary points generated by Theorem 1 at the center of their attractive regions.

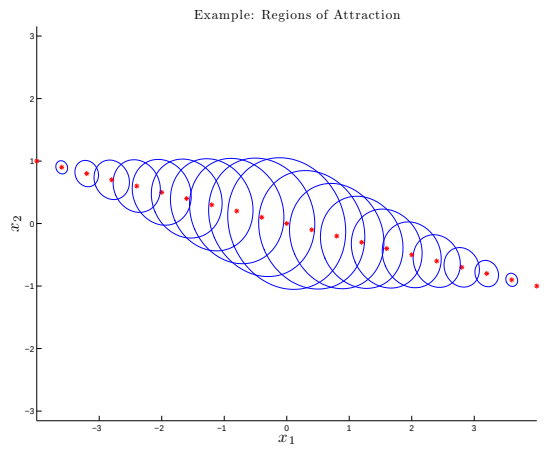


Fig. 2. An example of the regions of attraction for the given example system with $u_{max} = 1$.

B. Checking for Feasibility

We are interested in determining whether a given MDL program $(v_1, \epsilon_1), \ldots, (v_N, \epsilon_N)$ will successfully drive the system between the desired set-points under the input constraint $|u| \leq u_{max}$. It is clear that v_i must be in $\mathcal{V}$ for this to be possible, so we first make the following assumption:

Assumption 1: $v_i \in \mathcal{V},\ i = 1, \ldots, N$.

Now, in order for a MDL string to be feasible, the i^{th} set-point must lie in region of attraction for the $(i+1)^{th}$ set-point, and we state this fact as a lemma:

Lemma 2: Given an MDL string

$$\sigma = (v_1, \epsilon_1), \ldots, (v_N, \epsilon_N),$$

satisfying Assumption 1, let

$$\mathcal{B}_{\epsilon_i}(v_i) = \{x \in \mathbb{R}^n \mid (x - x_{v_i})^T P (x - x_{v_i}) \leq \epsilon_i\}$$

represent an ellipse around point x_{v_i}. If

$$\mathcal{B}_{\epsilon_i}(v_i) \subseteq \mathcal{E}(P, v_{i+1}) \tag{11}$$

then x can be transferred from x_{v_i} to $x_{v_{i+1}}$ while satisfying the bounded input constraint.
(Note here that the set $\mathcal{B}_{\epsilon_i}(v_i)$ is exactly the set where interrupt in equation (6) takes on the value 1.)

This lemma states that if the ellipse of size ϵ_i around point x_{v_i} is strictly within the region of attraction of $x_{v_{i+1}}$, then the system will arrive at $x_{v_{i+1}}$ (asymptotically) and satisfy the input constraints. Based on this pairwise characterization of the feasibility, we can now extend this notion to entire MDL strings:

Assumption 2:

$$x(0) \in \mathcal{E}(P, v_1).$$

Definition 1: The string

$$\sigma = (v_1, \epsilon_1), \ldots, (v_N, \epsilon_N),$$

is a *feasible program string* if it satisfies Assumptions 1 and 2, and

$$\mathcal{B}_{\epsilon_i}(v_i) \subseteq \mathcal{E}(P, v_{i+1}), \text{ for } i = 1, \cdots, N-1.$$

The set of these feasible program strings is denoted by $\mathcal{F}$.

We state the following theorem (whose rather obvious proof we omit):

Theorem 2 (Program feasibility): The MDL string σ drives the system close to the set-points (in the sense defined by the interrupts) if $\sigma \in \mathcal{F}$.

What this theorem gives us is a feasibility check for determining if the string does in fact perform as desired. However, if the feasibility condition is violated[1], something must be done, and in the next section, we discuss how to insert new control modes into the MDL string in order to respect the bounded input constraints yet drive through the desired intermediary set-points.

IV. Mode Insertion to Maintain Input Bounds

When an MDL string fails the feasibility check, we need to modify the string to ensure that the input constraints are satisfied. Our approach is to insert new modes into the string σ, so that the augmented string becomes a member of the feasible set $\mathcal{F}$. This method was inspired by *sequential composition*, as described in [3], by inserting modes when we know that the inserted region of attraction contains the a subset of the region of attraction of the previous mode.

Consider, again, the MDL string

$$\sigma = (v_1, \epsilon_1), \ldots, (v_N, \epsilon_N).$$

Each element in σ comes from a finite alphabet of modes, which we denote by $(v_i, \epsilon_i) \in \mathcal{A}$. The set of all possible concatenations of these elements is denoted by $\mathcal{A}^\star$; consequently, the each MDL comes from the set of all concatenations, i.e. $\sigma \in \mathcal{A}^\star$. We define the length operator as a mapping $\ell : \mathcal{A}^\star \to \mathbb{N}$, which accepts an MDL string and returns its number of elements. For example, if $\sigma = (v_1, \epsilon_1), (v_3, \epsilon_3)$, then $\ell(\sigma) = 2$.

Now, using our definitions, we state the mode insertion problem as:

$$\begin{aligned} & \min_{\sigma'} \ell(\sigma') \\ \text{s.t.} \quad & (v_i, \epsilon_i)\sigma'(v_{i+1}, \epsilon_{i+1}) \in \mathcal{F}. \end{aligned}$$

Thus, the problem of inserting intermediate points is characterized by minimizing the length of the string σ' such that the new string $(v_i, \epsilon_i)\sigma'(v_{i+1}, \epsilon_{i+1})$ is still in the set of feasible program strings, $\mathcal{F}$. This problem can be solved by inserting new modes, $(v_{k_l}, \epsilon_{k_l})$, such each pair of intermediate points satisfy the pairwise relation (11).

[1]Since our Lyapunov functions are *conservative* estimates of the region of attraction around each point, it may still be possible to execute an MDL string even if $\sigma \notin \mathcal{F}$.

A. MAXFORWARD *Algorithm*

We develop an algorithm, called MAXFORWARD, that builds up the intermediate MDL string σ' using the relation in (11). This algorithm begins with the starting element of the MDL string: (v_i, ϵ_i). When the system uses this MDL mode, the state is pulled toward the point $x_{v_i} \in \mathcal{X}$ until it crosses into the interrupt region, $\mathcal{B}_{\epsilon_i}(v_i)$, shown in Figure 3. From this region we need to insert a finite number of modes that can drive our system to $(v_{i+1}, \epsilon_{i+1})$.

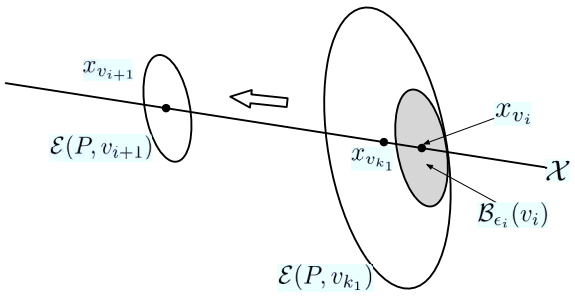


Fig. 3. MAXFORWARD

We want to choose a v_{k_l} such that $\mathcal{B}_{\epsilon_i}(v_i) \subseteq \mathcal{E}(P, v_{k_l})$. In other words, we need an ellipse that covers the interrupt region so that equation (11) in Lemma 2 holds. To find the value of v_{k_l} that results in the covering ellipse $\mathcal{E}(P, v_{k_l})$, we design an iterative algorithm that steps through possible values of $v \in \mathcal{V}$, starting at v_i. At each iteration we perform the update

$$v_{k_{j+1}} = v_{k_j} + \Delta v,$$

where $\Delta v > 0$ is a fixed step size. As the size of Δv increases the more numerical error enters into the insertion process, eventually creating incorrect mode insertions. If this increment results in $\mathcal{B}_{\epsilon_i}(v_{k_j}) \nsubseteq \mathcal{E}(P, v_{k_{j+1}})$ then the value v_{k_j} is chosen as an intermediate MDL mode: $(v_{k_j}, \epsilon_{k_j})$.

The first step in this process is visualized by the ellipse $\mathcal{E}(P, v_{k_1})$ covering $\mathcal{B}_{\epsilon_i}(v_i)$ in Figure 3. We repeat the algorithm until the m^{th} step, where

$$\mathcal{B}_{\epsilon_{k_m}}(v_{k_m}) \subseteq \mathcal{E}(P, v_{i+1}).$$

At this point we have the new intermediate string:

$$\sigma' = (v_{k_1}, \epsilon_{k_1}), \ldots, (v_{k_m}, \epsilon_{k_m}),$$

which maintains that $(v_i, \epsilon_i)\sigma'(v_{i+1}, \epsilon_{i+1}) \in \mathcal{F}$. Note that this algorithm produces an optimal path given our feasibility requirements. Without input bounds, the optimal solution would be more straightforward.

The formal algorithm is shown in listing Algorithm 1.

*Theorem 3 (*MAXFORWARD *Optimality):* If Algorithm 1 returns a solution σ' then it produces the minimal length string σ' such that $(v_i, \epsilon_i)\sigma'(v_{i+1}, \epsilon_{i+1}) \in \mathcal{F}$.

Proof: Assume Algorithm 1 returned the string σ' corresponding to m intermediate points $x_{v_1}, \cdots, x_{v_m}$ between x_{v_i} and $x_{v_{i+1}}$, i.e. Algorithm 1 inserted m new modes into the MDL string. We note that in order to go from x_{v_k} to $x_{v_{i+1}}$ (or more precisely, from small ellipses around these points) Algorithm 1 needs $m - k + 1$ modes.

Algorithm 1 MAXFORWARD Algorithm

```
Choose Δv > 0
v_{k_j} ← v_i {Initialize with first MDL mode's controller.}
covered ← FALSE
while B_{ε_{k_j}}(v_{k_j}) ⊈ E(P, v_{i+1}) do
  while ¬covered do
    v_{k_{j+1}} = v_{k_j} + Δv
    if B_{ε_{k_j}}(v_{k_j}) ⊈ E(P, v_{k_{j+1}}) then
      σ' ← (v_{k_j}, ε_{k_j}) {Add interim MDL mode.}
      covered ← TRUE
    end if
  end while
end while
return σ'
```

Now, since $v \in \mathbb{R}$, and hence $\dim(\mathbf{relint}(\mathcal{X})) = 1$, where **relint** denotes the relative interior, we can order points along $\mathcal{X}$ by how far away from $x_{v_{i+1}}$ they are. The first observation is that, by construction, the MAXFORWARD nature of Algorithm 1 prevents the existence of a point $x' \in \mathcal{X}$ such that x' can be reached in k or fewer steps from x_{v_i}, and $\|x' - x_{v_{i+1}}\| < \|x_{v_k} - x_{v_{i+1}}\|$. Instead, assume that σ' is *not* optimal and that the k^{th} intermediary point is $x'' \neq x_{v_k}$. Thus, we know that $\|x'' - x_{v_{i+1}}\| > \|x_{v_k} - x_{v_{i+1}}\|$.

But, the MAXFORWARD property again implies that no $x \in \mathcal{X}$ such that $\|x - x_{v_{i+1}}\| > \|x_{v_k} - x_{v_{i+1}}\|$ can reach $x_{v_{i+1}}$ in fewer steps than $m - k + 1$. As such, the optimal string (containing the intermediary point x'') can have no fewer elements than σ', and hence σ' is the (not necessarily unique) optimal solution. ■

This proof works because we have an order on the 1-dimensional space $\mathcal{X}$. If $\dim(\mathbf{relint}(\mathcal{X})) > 1$ then our algorithm and proof would have to consider multiple directions at each step due to the discretization of the state space.

B. Simulation Results

For our simulation results, we use the same choice of system matrices from the end of Section III-A. We specified a two mode MDL string: $(v_1, \epsilon)(v_2, \epsilon)$, where each mode uses the same sized interrupt region defined by $\mathcal{B}_\epsilon(v_i),\ i = 1, 2$ and the values of v_i come from the computed region $\mathcal{V}$.

Figure 4 shows the effort of the feedback controller as the system executes this MDL string. Once the system reaches the interrupt region of the first mode, the system switches to (v_2, ϵ), which causes a jump in the input signal that is well outside of the upper bound of the input constraint, $u_{max} = 1$. According to Definition 1, this MDL string is *not* feasible; hence, we must apply Algorithm 1 to insert intermediate modes.

The result of our MAXFORWARD algorithm, with $\Delta v = 0.001$, is shown in Figure 5 by the dotted ellipses. The algorithm inserted ten modes, allowing the system to move from its starting point x_0 to the final state x_f with bounded

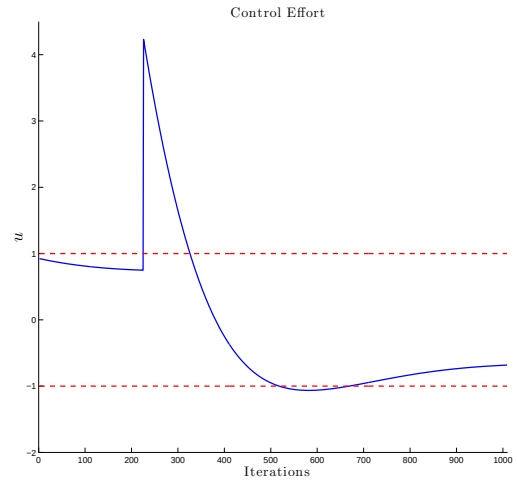


Fig. 4. The plot of the input u over the iterations of the simulation. Note that at the switch point, the system requires an input far greater than what the actuators can supply. The system leaves the input bound, again, as the second mode is executed.

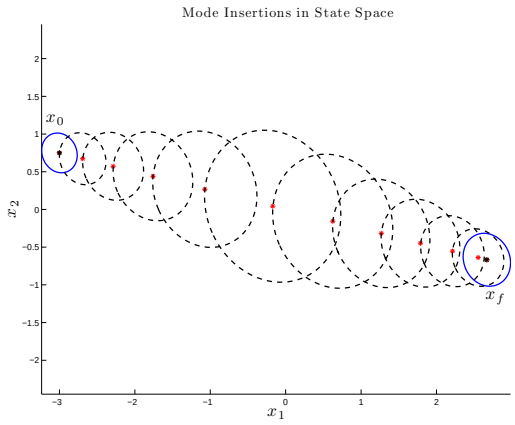


Fig. 5. This figure shows the ten new ellipses (dotted lines), $\mathcal{E}(P, v_{k_j})$ for $j = 1, \cdots, 10$, produced by the intermediate modes inserted into the MDL string.

control effort, as shown in Figure 6. Using the expanded MDL string lengthens the time it takes for the system to approach x_f; however, our design goal of maintaining the input constraints was met.

V. Conclusions

In this paper, we considered the problem of expanding motion programs when the system must operate under bounded input constraints. We presented the concept of a feasible MDL string and an algorithm that modifies an infeasible string so that it can execute on a system without violating input constraints. Finally, we demonstrated the algorithm on a linear system and showed that the new MDL string satisfied the input constraints, while still moving the state to our desired destination.

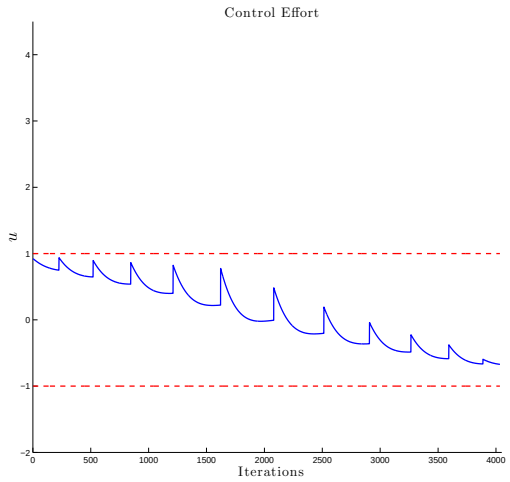


Fig. 6. In this plot, we see the successful maintenance of the control bound $|u| < 1$ during the execution of the expanded MDL string.

Acknowledgement

We acknowledge the U.S. National Science Foundation for its support through grant number 0820004. We also thank Hongyi Li and Amir Rahmani for helpful discussions.

References

[1] R.W. Brockett. On the Computer Control of Movement. In the *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, pp. 534–540, New York, April 1988.

[2] R.R. Burridge, A.A. Rizzi, and D.E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, Vol. 8, No. 6, pp. 534-555, June 1999.

[3] M. Egerstedt and R.W. Brockett. Feedback can reduce the specification complexity of motor programs. *IEEE Transactions on Automatic Control*, Vol. 48, No. 2, pp. 213–223, Feb. 2003.

[4] H. Krishnan and M. Vidyasagar. Bounded input feedback control of linear systems with application to the control of a flexible beam. In *Proceedings of the 27^{th} IEEE Conference on Decision and Control*, December, 1988.

[5] I. Kolmanovsky and E.G. Gilbert. Multimode Regulators for Systems with State and Control Constraints and Disturbance Inputs. In A. Morse (Ed.) *Control Using Logic-Based Switching*, Springer, 1997.

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

[7] V. Manikonda, P. S. Krishnaprasad, and J. Hendler. Languages, behaviors, hybrid architectures and motion control. In J.C. Willems and J. Baillieul (Eds.) *Mathematical Control Theory*, Springer-Verlag, 1998.

[8] P. Martin, J.P. de la Croix, and M. Egerstedt. MDLn: A Motion Description Language for Networked Systems. In *Proceedings of 47^{th} IEEE Conference on Decision and Control*, December 2008.

[9] M.W. McConley, B.B. Appleby, M.A. Dahleh, and E. Feron. A computationally efficient Lyapunov-based scheduling procedure for control of nonlinear systems with stability guarantees. *IEEE Transactions on Automatic Control*, Vol. 45, No. 1, January 2000.

[10] G.N. Saridis and Z.V. Rekasius. Investigation of worst-case errors when inputs and their rate of change bounded. *IEEE Transactions on Automatic Control*, Vol. 11, pp. 266-300, April 1966.

[11] R. Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems*, 2009.

[12] G.F. Wredenhagen and P.R. Belanger. Piecewise-linear LQ control for systems with input constraints. *Automatica*, Vol. 30, No. 3, pp. 403–416, 1994.