An Optimal Control Approach to Mode Generation in Hybrid Systems*

Tejas R. Mehta and Magnus Egerstedt
{tmehta,magnus}@ece.gatech.edu
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA

July 25, 2005

Abstract

This paper presents a solution to the problem of constructing control programs, i.e. sequences of control modes, from a given motion alphabet. In particular, techniques are developed that enable reinforcement learning to act directly at the mode level, and hence make learning applicable to continuous time control systems in a straight-forward manner. Moreover, given such a control program, the issue of improving the system performance through the addition of new control laws is addressed as an optimal control problem. In fact, this is achieved through an optimal combination of recurring mode strings. A number of examples are provided that illustrate the viability of the proposed methods.

1 Introduction

In order to manage the complexity associated with many modern control applications, multi-modal control has emerged as a viable approach in which a number of control modes are constructed and combined according to some supervisory control logic, e.g. [13, 8, 22, 25]. The idea is that the design of each individual mode, designed with respect to a particular control task, data source, operating point, or system configuration, constitutes a significantly more manageable task than the design of one single, multi-objective control law. A prime example of this design paradigm is behavior based robotics (for example, see [1]), in which the overall robot behavior is obtained through a combination of simpler behaviors such as avoiding obstacles or approaching landmarks.

This divide and conquer strategy implies that the main design focus is shifted from the problem of producing control laws, i.e. input-to-output maps, to the

*This work was supported by DARPA through Grant number FA8650-04-C-7131

problem of concatenating or combining modes. Moreover, if one thinks of the set of available behaviors as an alphabet, the complexity of this set can be identified with its cardinality (possibly in combination with the length of the mode strings needed to achieve the control objective), while its expressiveness can be thought of as a measure of what overall behaviors are producible from the set. One can thus ask the question of whether or not it is worth adding new modes to the mode set, i.e. to try to weigh the increase in complexity against the potential increase in expressiveness.

In this paper, we will address both of these aforementioned issues:

1. Given a mode set, how should the modes be concatenated in order to achieve a desirable, overall performance?

2. Given a mode set, how should it be augmented in order to improve the overall behavior of the system?

We will address both of these issues from the vantage point of optimal control. In particular, a reinforcement learning algorithm will be proposed that acts at the mode level for producing the control programs, while the Calculus of Variations will be employed in order to augment the mode set as a function of the current modes.

The outline of this paper is as follows: In Section 2, standard reinforcement learning techniques are recalled. It is moreover shown how these techniques can be augmented to support the learning of control programs, operating on continuous time systems whose state spaces are differentiable manifolds. This is in contrast to the standard reinforcement learning methods defined for discrete time systems with finite input-output sets. In Section 3 we discuss the issue of adding new control laws to the mode set and illustrate the basic idea with examples for both discrete and continuous time linear systems. Section 4 presents the general framework in which the Calculus of Variations is used for producing new modes from a given motion alphabet, followed by the conclusions in Section 5.

2 Learning Multi-Modal Control Programs

For systems operating in unknown environments and/or with unknown dynamics, reinforcement learning provides the means for systematic trial-and-error interactions with the environment. In this section, we will, first, briefly cover the standard reinforcement-learning model commonly used for discrete-time systems with finite state and control spaces. Then, we will show how to apply learning techniques to multi-modal hybrid systems in order to facilitate learning multi-modal control programs for continuous-time systems.

2.1 Standard Reinforcement Learning

In the standard reinforcement-learning model, at each step (discrete time), the agent chooses an action, $u \in U_F$, based on the current state, $x \in X_F$, of the

environment, where U_F and X_F are finite sets (Hence the subscript F). The corresponding result is given by $x_{k+1} = \delta(x_k, u_k)$, where $\delta : X_F \times U_F \to X_F$ is the state transition function that encodes the system dynamics. Moreover, a cost $c : X_F \times U_F \to \mathbb{R}$ is associated with taking action u at state x. The agent should choose actions in order to minimize the overall cost. Given a policy $\pi : X_F \to U_F$, the discounted cost that we wish to minimize is given by

$$V^{\pi}(x_0) = \sum_{k=0}^{\infty} \gamma^k c(x_k, \pi(x_k)), \tag{1}$$

where $\gamma \in (0, 1)$ is the discount factor and $x_{k+1} = \delta(x_k, \pi(x_k))$, $k = 0, 1, \ldots$

We will use $V^*(x)$ to denote the minimum discounted cost incurred if the agent starts in state x and executes the optimal policy, denoted by π^*. In other words, the optimal value function is defined through the Bellman equation

$$V^*(x) = \min_{u \in U_F} \big[c(x, u) + \gamma V^*(\delta(x, u))\big], \forall x \in X_F. \tag{2}$$

Equation (2) simply states that the optimal value is obtained by taking the action that minimizes the instantaneous cost plus the remaining discounted cost. Once V^* is known, the optimal policy, π^*, follows directly through

$$\pi^*(x) = \min_{u \in U_F} \big[c(x, u) + \gamma V^*(\delta(x, u))\big], \tag{3}$$

which shows why knowing V^* is equivalent to knowing the optimal policy.

If we now let $Q^*(x, u)$ be the discounted cost for taking action u in state x and then continuing to act optimally, we observe that $V^*(x) = \min_u Q^*(x, u)$, and therefore

$$Q^*(x, u) = c(x, u) + \gamma \min_{u' \in U_F} Q^*(\delta(x, u), u'). \tag{4}$$

To find Q^*, we start by assigning a uniform value to every state-action pair, and then randomly select state-action pairs (x, u) and update the Q-table using the following Q-learning law

$$Q_k(x, u) := Q_{k-1}(x, u) + \alpha_k \Big(c(x, u) + \gamma \min_{u' \in U_F} \Big\{ Q_{k-1}(\delta(x, u), u') - Q_{k-1}(x, u) \Big\} \Big). \tag{5}$$

If each action is selected at each state an infinite number of times on an infinite run and α_k, the learning rate, is decayed appropriately, the Q values will converge to Q^* with probability 1. By appropriate decay of α_k we mean that $\sum_k \alpha_k = \infty$ while $\sum_k \alpha_k^2 < \infty$, hence decreasing the learning rate over time (e.g. $\alpha_k = 1/k$) will guarantee convergence. (For more details regarding reinforcement learning, see for example [17, 19, 29, 32, 33].)

2.2 Learning Control Programs for Discrete-Time Systems

We now define a new input space that corresponds to tokenized descriptions of feedback laws and interrupts, as prescribed within the motion description

language (MDL) framework [11, 13, 12, 16]. Instead of interacting with the environment at each step, the agent takes actions based on a feedback law κ, which is a function of the state x. The agent furthermore continues to act on the feedback control law κ until the interrupt ξ triggers, at which point a scalar cost is incurred.

Formally, let X_F and U_F be finite sets, as defined earlier, and let $\Sigma = K \times \Xi$, where $K \subseteq {U_F}^{X_F}$ (the set of all maps from X_F to U_F) and $\Xi \subseteq \{0,1\}^{X_F}$. Moreover, let $\tilde{\delta} : X_F \times \Sigma \to X_F$ be the state transition mapping, $\tilde{x}_{k+1} = \tilde{\delta}(\tilde{x}_k, (\kappa_k, \xi_k))$, obtained through the following free-running, feedback mechanism [13]: Let $\tilde{x}_0 = x_0$ and evolve x according to $x_{k+1} = \delta(x_k, \kappa_0(x_k))$ until the interrupt triggers, i.e. $\xi_0(x_{k_0}) = 1$ for some index k_0. Now let $\tilde{x}_1 = x(k_0)$ and repeat the process, i.e. $x_{k+1} = \delta(x_k, \kappa_1(x_k))$ until $\xi_1(x_{k_1}) = 1$. Now let $\tilde{x}_2 = x(k_1)$, and so on. Also let $\zeta : X_F \times \Sigma \to \mathbb{R}$ be the cost associated with the transition.

We want to apply reinforcement learning to this model. To accomplish this we must make a few modifications. First, note that $card(\Sigma)$ is potentially much larger than $card(U_F)$, where $card(\cdot)$ denotes the cardinality. This directly affects the number of entries in our Q-table. If all possible feedback laws and interrupts were available, the cardinality of the new input space would be $[2card(U_F)]^{card(X_F)}$ with obvious implications for the numerical tractability of the problem.

Second, in order to find Q^*, we start again by assigning a uniform value to every state-action pair, and then iteratively update the Q values by randomly selecting a state-action pair with the action comprising of one of the possible feedback laws in K and interrupts in Ξ. The consequent Q-learning law is

$$Q_k(x,(\kappa,\xi)) := Q_{k-1}(x,(\kappa,\xi)) + \alpha_k\Big(\zeta(x,(\kappa,\xi)) + \\ +\gamma \min_{(\kappa',\xi')}\Big\{Q_{k-1}(\tilde{\delta}(x,(\kappa,\xi)),(\kappa',\xi')) - Q_{k-1}(x,(\kappa,\xi))\Big\}\Big). \quad (6)$$

Since Ξ and K are finite, the set of all possible modes Σ is finite as well. Hence the convergence results still hold, as long as each mode is selected for each state an infinite number of times, and α_k decays appropriately.

2.3 Learning Control Programs for Continuous-Time Systems

Now that the discrete-time case with finite state and input spaces is covered, we shift focus to the problem of learning multi-modal control programs for continuous-time systems. Suppose we have the following system:

$$\dot{x} = f(x,u), \quad \text{where } x \in X = \mathbb{R}^n, u \in U = \mathbb{R}^m, \text{ and } x(t_0) = x_0 \text{ is given.} \quad (7)$$

If at time t_0, the system receives the input string $\sigma = (\kappa_1,\xi_1),\ldots,(\kappa_q,\xi_q)$, where $\kappa_i : X \to U$ is the feedback control law, and $\xi_i : X \to \{0,1\}$ is the interrupt,

then x evolves according to

$$\begin{aligned} \dot{x} &= f(x, \kappa_1(x)); \quad t_0 \leq t < \tau_1 \\ &\vdots \qquad\qquad\quad \vdots \\ \dot{x} &= f(x, \kappa_q(x)); \quad \tau_{q-1} \leq t < \tau_q, \end{aligned}$$

where τ_i denotes the time when the interrupt ξ_i triggers (i.e. changes from 0 to 1).

We are interested in finding a sequence of control-interrupt pairs that minimizes a given cost for such a system. For example, we might be interested in driving the system to a certain part of the state space (e.g. to the origin), and penalize the final deviation from this target set. Previous work on reinforcement learning for continuous-time control systems can broadly be divided into two different camps. The first camp represents the idea of a direct discretization of the temporal axis as well as the state and input spaces (e.g. [7, 28]). The main criticism of this approach is that if the discretization is overly coarse, the control optimizing the discretized problem may not be very good when applied to the original problem. Of course, this complication can be moderated somewhat by making the discretization more fine. Unfortunately, in this case, the size of the problem very quickly becomes intractable.

The second approach is based on a temporal discretization (sampling) in combination with the use of appropriate basis functions to represent the Q-table (e.g. [10, 24, 29]). Even though this is a theoretically appealing approach, it lacks in numerical tractability. In contrast to both these two approaches, we propose to let the temporal quantization be driven by the interrupts directly (i.e. not by a uniform sampling) and let the control space have finite cardinality through the interpretation of a control symbol as a tokenized control-interrupt pair. In other words, by considering a finite number of feedback laws $\kappa_i : X \to U,\ i = 1, \ldots, M$, together with interrupts $\xi_j,\ j = 1, \ldots, N$, the control space (viewed at a functional level) is finite even though the actual control signals take on values in $\mathbb{R}^m$. Another effect of the finite mode-set assumption is that it provides a natural quantization of the state space. Moreover, if we bound the length of the mode sequences, this quantization is in fact resulting in a finite set of reachable states.

Given an input $\sigma = (\kappa, \xi) \in \Sigma$, where $\Sigma \subseteq U^X \times \{0, 1\}^X$, the flow is given by

$$\phi(x_0, \sigma, t) = x_0 + \int_0^t f(x(s), \kappa(x(s)))ds. \tag{8}$$

If there exists a finite time $T \geq 0$ such that $\xi(\phi(x_0, \sigma, T)) = 1$, then we let the interrupt time be given by

$$\tau(\sigma, x_0) = min\{t \geq 0 \mid \xi(\phi(x_0, \sigma, t)) = 1\}. \tag{9}$$

If no such finite time T exists then we say that $\tau(\sigma, x_0) = \tau_\infty$ for some distinguishable symbol τ_∞. Furthermore, we let the final point on the trajectory

generated by σ be

$$\chi(\sigma, x_0) = \phi(x_0, \sigma, \tau(\sigma, x_0))$$

if $\tau(\sigma, x_0) \neq \tau_\infty$ and use the notation $\chi(\sigma, x_0) = \chi_\infty$ otherwise. Moreover let $\chi(\sigma, \chi_\infty) = \chi_\infty, \forall \sigma \in \Sigma$.

This construction allows us to define the Lebesque sampled finite state machine $(X_N^Q, \Sigma, \tilde{\delta}, \tilde{x_0})$, where N is the longest allowable mode string, and where the state transition is given by

$$\begin{aligned}
&\tilde{x}_0 = x_0 \\
&\tilde{x}_{k+1} = \tilde{\delta}(\tilde{x}_k, \sigma_k) = \chi(\sigma_k, \tilde{x}_k), k = 0, 1, \ldots
\end{aligned}$$

The state space X_N^Q is given by the set of all states that are reachable from $\tilde{x}_0$ using mode strings of length less than or equal to N.

Now that we have a finite state machine describing the dynamics, we can directly apply the previously discussed reinforcement learning algorithm, with an appropriate cost function, in order to obtain the optimal control program. However, in order to preserve computing resources, we run this in parallel with the state exploration, and the general algorithm for accomplishing this is given by

$\mathcal{X} := \{\tilde{x}_0, \tilde{\delta}(\tilde{x}_0, \sigma)\},\ \forall \sigma \in \Sigma$
$step(\tilde{x}_0) := 0$
$step(\tilde{\delta}(\tilde{x}_0, \sigma)) := 1,\ \forall \sigma \in \Sigma$
$p := 1$
$Q_p(\tilde{x}, \sigma) := const\ \forall \tilde{x} \in \mathcal{X}, \sigma \in \Sigma$
`repeat`
  $p := p + 1$
  $\tilde{x} := rand(\chi \in \mathcal{X} \mid step(\chi) < N)$
  $\sigma := rand(\Sigma)$
  $\tilde{x}' := \tilde{\delta}(\tilde{x}, \sigma)$
  `if` $\tilde{x}' \notin \mathcal{X}$ `then`
    $step(\tilde{x}') := step(\tilde{x}) + 1$
    $\mathcal{X} := \mathcal{X} \cup \{\tilde{x}'\}$
    $Q(\tilde{x}', \sigma) := const\ \forall \sigma \in \Sigma$
  `end if`
  $Q_p(\tilde{x}, \sigma) := Q_{p-1}(\tilde{x}, \sigma)$
    $+ \alpha_p\Big(\zeta(\tilde{x}, \sigma) + \gamma \min_{\sigma' \in \Sigma}\Big\{Q_{p-1}(\tilde{x}', \sigma') - Q_{p-1}(\tilde{x}, \sigma)\Big\}\Big)$
`until` $mod(p, L) = 0$ *and* $|Q_p(\tilde{x}, \sigma) - Q_{p-L}(\tilde{x}, \sigma)| < \epsilon,\ \forall\ \tilde{x} \in \mathcal{X}, \sigma \in \Sigma$
$X_N^Q = \mathcal{X}$

Unlike the earlier Q-learning algorithm, the state space is initially unknown for this case, and we thus begin learning/exploring from the states we know (namely $\tilde{x}_0$ and all the states reachable in one step). At each iteration of the learning process, we select a state randomly from the set of known states and we select a mode randomly from the set of modes. In the algorithm, the function

$step(\tilde{x})$ represents the length of the shortest control program used so far to reach state $\tilde{x}$ from the initial state $\tilde{x}_0$. This is to ensure we only explore states that are reachable from $\tilde{x}_0$ using mode strings of length less than or equal to N, i.e. $\mathcal{X} \subseteq X_N^Q$. We then calculate the next state and determine if it is a member of our known state space (In practice, it is necessary to check if the next state belongs to a neighborhood of a previously visited state). If not, add this state to the known state space and make the corresponding change in the Q-table. We continue to explore and update the state space and our Q-table (or value function) in this manner until the Q-table is stationary. Note that in the algorithm, $\epsilon > 0$ is a small positive scalar and L is a large number needed to ensure that sufficiently many state-action pairs are visited.

In summary, the algorithm above effectively estimates the reachable set, while applying reinforcement learning over this estimated set in order to obtain the optimal mode sequence. An example of this approach is shown in Figure 1. Here we find the optimal path to drive a unicycle from x_0 to x_g while avoiding obstacle and minimizing the length of the control sequence along with the total distance travelled. The dynamics for the unicycle are

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega. \end{aligned} \tag{10}$$

In the system above (x, y) are the Cartesian coordinate of the center of the unicycle and ϕ is its orientation with respect to the x-axis. Assume that v is constant and ω is the control variable. The system has three behaviors, namely "go-to-goal", "avoid-obstacle", and "combo", which is the combination of the two previous behaviors. Given these set of behaviors and the initial state x_0, the thin lines represent the reachable trajectories along with the estimated reachable set, which is given by the 'dot' that terminates these trajectories. The thick line is the optimal learned path to drive the unicycle for x_0 to x_g given these set of modes and the performance criterion described earlier.

3 Motion Alphabet Augmentation

Now that we have a method in which strings of control modes, i.e. control programs, can be produced from a given mode set, one natural question to ask is whether or not it is beneficial to add new modes to the mode set. Moreover, one would like to change the control programs without having to recompute the entire mode sequence, e.g. through the previously discussed reinforcement learning method. If we let Σ be the set of available modes and let the mode string $\bar{\sigma} = \sigma_1, \sigma_2, \ldots, \sigma_q$ solve a particular control task, e.g. one that drives the system to the origin, one can define the complexity of this control program as the number of bits needed for its encoding, as was the case in [13, 11]. In other words, the complexity of the control program $\bar{\sigma}$, whose elements are drawn from the mode set Σ, is given by

$$length(\bar{\sigma}) \log_2(card(\Sigma)),$$

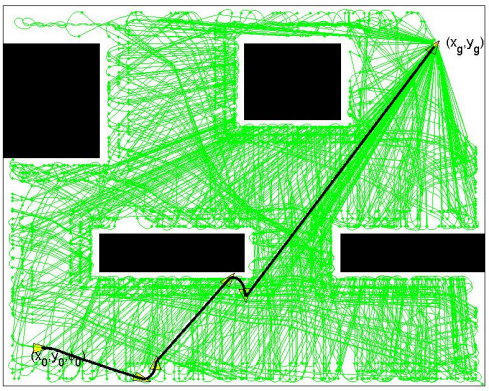


Figure 1: Estimated reachable set along with the optimal learned path (thick) to drive a unicycle from x_0 to x_g given a set of modes.

where $length()$ denotes the number of modes in the string, and as before $card()$ denotes cardinality.

Now, we are interested in producing new modes in a highly structured manner so that these new modes are functions of the old modes. Moreover, the mode string should be readily updated to account for these new modes without incurring any hefty computational costs. Our proposed solution is based on the observation that if a given mode string $\sigma_1, \ldots, \sigma_r$ occurs (possibly repeatedly) in the control program, it might be possible and beneficial to replace this string with one single mode σ_{1r} that results in (roughly) the same behavior. If p occurrences of $\sigma_1, \ldots, \sigma_r$ are replaced by σ_{1r} in the original control program ($\bar{\sigma}_{old}$), then the complexity of new mode string ($\bar{\sigma}_{new}$) is now

$$(length(\bar{\sigma}_{old}) - p(r-1))card(\Sigma_{old} + 1) < length(\bar{\sigma}_{old})card(\Sigma_{old})$$

as long as $p \geq 1$ and $r > 1$. In other words, the complexity would be reduced if such a σ_{1r} could be found. The construction of such new, replacement modes is the topic of the next sections.

However, before deriving a general framework and algorithms for adding new modes by replacing recurring mode sequences by one single mode, we will look at some specific, motivating examples. In particular, we will examine the case of the continuous and discrete time linear time-invariant (LTI) systems.

3.1 Discrete Time LTI Systems

Consider the following discrete-time system:

$$x_{k+1} = Ax_k + Bu_k, \quad \text{where } x \in \mathbb{R}^n \text{ and } u \in \mathbb{R}^m. \tag{11}$$

Suppose we have a set of q modes given by linear feedback mappings $\kappa_i : X \to U$, say $\kappa_i(x_k) = -K_i x_k$, for $i = 1, 2, \ldots, q$, and interrupts which trigger after a given number of steps. We let ξ_{n_j} denote the interrupt that triggers after $n_j \in \mathbb{Z}_+$ steps. Suppose the system given in (11) starts from x_0 and advances using the input mode sequence $\bar{\sigma} = \sigma_1\sigma_2\cdots\sigma_l$, where $\sigma_j = (\kappa_j, \xi_{n_j})$. Then the evolution of the system is given as follows:

$$\begin{aligned}
x_1 &= Ax_0 - BK_1x_0 = (A - BK_1)x_0 \\
x_2 &= (A - BK_1)x_1 = (A - BK_1)^2x_0 \\
&\vdots \\
x_{n_1} &= (A - BK_1)^{n_1}x_0 \\
x_{n_1+1} &= (A - BK_2)x_{n_1} = (A - BK_2)(A - BK_1)^{n_1}x_0 \\
&\vdots \\
x_{n_1+n_2} &= (A - BK_2)^{n_2}(A - BK_1)^{n_1}x_0 \\
&\vdots \\
x_f \equiv x_{n_1+\ldots+n_l} &= \prod_{i=1}^{l}(A - BK_i)^{n_i}x_0
\end{aligned}$$

Now suppose we are allowed to add a new mode, $(\kappa_{q+1}, \xi_{q+1})$, where the feedback law has to be a function of the existing modes, i.e. $\kappa_{q+1} = \delta(\kappa_1, \kappa_2, \ldots, \kappa_q)$, in order to improve performance. In an attempt to decrease the size of the control program, we propose to replace the entire mode sequence $\bar{\sigma}$ with one new mode, i.e. select $(\kappa_{q+1}, \xi_{n_q+1})$ such that

$$\tilde{x}_f = (A - BK_{q+1})^{n_{q+1}}x_0 \approx x_f.$$

Thus the problem is to find a new feedback law $\kappa_{q+1}(x) = -K_{q+1}x$ and the interrupt $\xi_{n_{q+1}}$, while minimizing some given performance criterion. In the following sections, we look at two variations of the problem. First we will setup and solve

$$P1 : \min_{\kappa_{q+1}} \; J_1 = \| \, x_f - \tilde{x}_f \, \|^2,$$

e.g. we solve this problem locally as a function of the initial state x_0. Next we analyze

$$P2 : \min_{\kappa_{q+1}} \; J_2 = \| \, \Phi_q(n_q)\cdots\Phi_2(n_2)\Phi_1(n_1) - \Phi_{q+1}(n_{q+1}) \, \|_F^2,$$

where $\Phi_i(n) = (A - BK_i)^n$ for $i = 1, 2, \ldots, q+1$. Here we try to find a global solution, independent of the initial state x_0, to the problem of replacing a mode sequence with one new mode. Moreover, in both of these problems we assume that $\xi_{n_{q+1}} = \xi_1$ and let δ be a linear function so that $K_{q+1} = \sum_{i=1}^{q} \alpha_i K_i$. Hence both problems deal with finding the coefficient vector $\vec{\alpha}$ such that the cost $J(\vec{\alpha})$ is minimized.

3.1.1 Problem 1

Formally we wish to,

$$\begin{aligned}
\min_{\vec{\alpha}} J_1 &= \min_{\vec{\alpha}} \parallel x_f - \tilde{x}_f \parallel^2 \\
&= \min_{\vec{\alpha}} \parallel x_f - (A - \sum_{i=1}^{q} \alpha_i BK_i)x_0 \parallel^2 \\
&\equiv \min_{\vec{\alpha}} \parallel c + \sum_{i=1}^{q} \alpha_i BK_i x_0 \parallel^2, \qquad (12)
\end{aligned}$$

where $c = x_f - Ax_0 \in \mathbb{R}^n$. Now differentiating (12) with respect to α_j and setting it equal to 0 to obtain the first-order necessary conditions, we obtain

$$\frac{\partial J_1}{\partial \alpha_j} = 2c^T x_0 K_j B + 2x_0^T K_j^T B^T \sum_{i=1}^{q} \alpha_i^* BK_i x_0 \equiv 0, \text{ for } j = 1, 2, \ldots, q. \qquad (13)$$

Here α_i^* is the optimal i^{th} coefficient. Dividing (13) by 2, and letting $m_j = BK_j x_0 \in \mathbb{R}^n$, we obtain

$$\begin{bmatrix} m_1^T m_1 & m_1^T m_2 & \ldots & m_1^T m_q \\ m_2^T m_1 & m_2^T m_2 & \ldots & m_2^T m_q \\ \vdots & \vdots & \ddots & \vdots \\ m_q^T m_1 & m_q^T m_2 & \ldots & m_q^T m_q \end{bmatrix} \vec{\alpha}^* = - \begin{bmatrix} c^T m_1 \\ c^T m_2 \\ \vdots \\ c^T m_q \end{bmatrix}. \qquad (14)$$

If we rewrite (14) as $M\vec{\alpha}^* = b$, then clearly $\vec{\alpha}^* = M^{-1}b$ if M is invertible.

As we will see, M derived above is invertible if and only if (a) $n \geq q$ and (b) $m_i \neq \sum_{j \neq i} \gamma_j m_j$ for $i = 1, 2, \ldots, q$ and any real coefficients γ_j. This fact follows from the fact that M is symmetric and has a lot of additional structure. To see this, define the matrix $\bar{M} = [m_1, m_2, \cdots, m_q] \in \mathbb{R}^{n \times q}$, then the Gram matrix $G(\bar{M})$ is the matrix of inner products of the vectors in $\bar{M}$, i.e. ij^{th} element of $G(\bar{M})$ is given by $< m_i, m_j >$. If we let $< m_i, m_j >= m_i^T m_j$, then $G(\bar{M}) = M$. Then the *grammian* of matrix $\bar{M}$ (written $gram(\bar{M})$) is defined to be the determinant $\det(G(\bar{M}))$, which in fact is equal to $\det(M)$. The grammian has several useful properties:

1. $gram(\bar{M})$ is real and ≥ 0.
2. $gram(\bar{M}) > 0$ if and only if the columns of X are linearly independent.

3. If $\bar{M} \in \mathbb{R}^{n \times q}$, then $gram(\bar{M}) = 0$ if $n < q$.

The last two of these properties readily provide the desired invertibility conditions.

Its clear that (14) has an unique solution if M is invertible ($\det(M) \neq 0$) and either infinitely many solutions or no solution if M is not invertible. However, as we will show, the problem will always have at least one solution. Hence, we can always replace the mode string $\bar{\sigma}$ with σ_{q+1} while minimizing the error $\parallel x_f - \tilde{x}_f \parallel^2$. Moreover, the error is zero if there are n independent basis vectors m_i's from the q available modes, where $m_i = BK_i x_0$. Suppose we have zero independent basis vectors from the q available modes (i.e. $m_i = \vec{0}$ for all q modes), then the system is autonomous (i.e. $x_{k+1} = Ax_k$ for all modes). Hence, any combination of the q modes will result in the same trajectory and the problem has infinitely many solutions. Now if there are k independent basis vectors m_i's (where $k \in \{1, 2, ..., n\}$), then we can let the new mode be a linear combination of the k modes corresponding to these k independent basis vectors. In this case the matrix M defined above will be invertible as noted above, and this subproblem of replacing the mode string as a combination of k modes has an unique solution.

However, the solution to the original problem of obtaining a new mode as a linear combination of the q given modes may not be unique. Since m_i is a n-dimensional vector, we cannot have more than n independent basis vectors. Hence we have covered all the cases, and the problem always has a solution. Moreover if we have n independent basis, these basis can span $\mathbb{R}^n$, and thus any state x_f is reachable with an appropriate choice of control vector $\vec{\alpha}$. Note that

$$\begin{aligned} \tilde{x}_f = (A - BK_{q+1})x_0 &= Ax_0 - \alpha_1^* BK_1 x_0 - \ldots - \alpha_n^* BK_n x_0 \\ &= Ax_0 - \alpha_1^* m_1 - \ldots - \alpha_n^* m_n \end{aligned}$$

So clearly if all m_i's are independent, and $\vec{\alpha}^*$ chosen to minimize the error $||x_f - \tilde{x_f}||$, we obtain $\tilde{x}_f = x_f$, and the error is zero.

As an example of using this approach to replace a mode sequence $\bar{\sigma}$ with a new mode, consider the system given in (11), with $n = 3$ and $m = 3$. Suppose we have a set of five feedback mappings $\kappa_i(x_k) = -K_i x_k$, for $i = 1, 2, \ldots, 5$. Here we wish to replace the mode sequence $\bar{\sigma} = (\kappa_1, \xi_1)(\kappa_2, \xi_1) \cdots (\kappa_5, \xi_1)$ with a new mode $\sigma_6 = (\kappa_6, \xi_1)$, where $\kappa_6(x) = K_6 x = \sum_{i=1}^{5} \alpha_i K_i x$. As noted earlier ξ_{n_j} is an interrupt that triggers after n_j steps. Using the technique given above, we compute the M matrix sequentially until we have three independent basis vectors m_i's (since $n = 3$). Next we obtained $\vec{\alpha}$, and thus the new control law κ_6. The results are shown in Figure 2, where the solid line represents the original trajectory of x obtained using the control string $\bar{\sigma}$, while the dashed line shows the trajectory of $\tilde{x}$ obtained using the new mode (here the control string is σ_6). Since we were able to find three independent basis, the error is zero as shown in the figure.

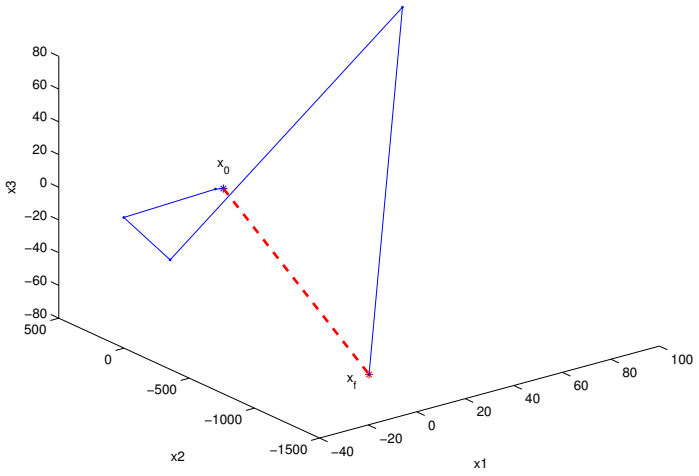


Figure 2: The solid line represents the trajectory of x, while the dashed line shows the trajectory of $\tilde{x}$ obtained using the new mode σ_6.

3.1.2 Problem 2

The solution presented in the preceding section depends on the initial condition x_0. In this section, we change the performance criterion to remove this dependence on x_0. Before stating this criterion, let $\Phi_i(n) = (A - BK_i)^n$ denote the state transition matrix associated with using mode (κ_i, ξ_n). Also the Forbenius norm of a matrix A is denoted $\| A \|_F = \sqrt{Tr(AA^H)}$, where A^H is the conjugate transpose and $Tr(.)$ is the matrix trace. So formally we wish to,

$$\begin{aligned}
\min_{\vec{\alpha}} J_2 &= \min_{\vec{\alpha}} \| \Phi_q(n_q)\Phi_{q-1}(n_{q-1}) \cdots \Phi_1(n_1) - \Phi_{q+1}(n_1) \|_F^2 \\
&= \min_{\vec{\alpha}} \| \Phi_q(n_q)\Phi_{q-1}(n_{q-1}) \cdots \Phi_1(n_1) - (A - \sum_{i=1}^{q} \alpha_i BK_i) \|_F^2 \\
&\equiv \min_{\vec{\alpha}} \| C + \sum_{i=1}^{q} \alpha_i BK_i \|_F^2, \qquad (15)
\end{aligned}$$

where $C = \Phi_q(n_q)\Phi_{q-1}(n_{q-1}) \cdots \Phi_1(n_1) - A \in \mathbb{R}^{n \times n}$. Now differentiating (15) with respect to α_j setting it equal to 0 to obtain the first-order necessary conditions, we obtain

$$\frac{\partial J_2}{\partial \alpha_j} = 2Tr(C^T BK_j) + 2 \sum_{i=1}^{q} \alpha_i^* Tr(BK_j K_i^T B^T) \equiv 0, \text{ for } j = 1, 2, \ldots, q. \quad (16)$$

Here we used the some of the common matrix trace properties, which include $Tr(A) = Tr(A^T)$, $Tr(A + B) = Tr(A) + Tr(B)$, and $Tr(\alpha A) = \alpha Tr(A)$. Di-

viding (16) by 2, and letting $N_j = BK_j \in \mathbb{R}^{n\times n}$, we obtain

$$\begin{bmatrix} Tr(N_1^T N_1) & Tr(N_1^T N_2) & \dots & Tr(N_1^T N_q) \\ Tr(N_2^T N_1) & Tr(N_2^T N_2) & \dots & Tr(N_2^T N_q) \\ \vdots & \vdots & \ddots & \vdots \\ Tr(N_q^T N_1) & Tr(N_q^T N_2) & \dots & Tr(N_q^T N_q) \end{bmatrix} \vec{\alpha}^* = - \begin{bmatrix} Tr(C^T N_1) \\ Tr(C^T N_2) \\ \vdots \\ Tr(C^T N_q) \end{bmatrix}. \quad (17)$$

If we rewrite (17) as $N\vec{\alpha}^* = b$, then clearly $\vec{\alpha}^* = N^{-1}b$ if N is invertible.

Similarly to $P1$, we find that N derived above is invertible if and only if (a) $n^2 \geq q$ and (b) $N_i \neq \sum_{j\neq i} \gamma_j N_j$ for $i = 1, 2, \dots, q$ and any real coefficients γ_j. To see this, we note that the trace is a valid inner product in $\mathbb{R}^{n\times n}$. So we can define the Gram matrix as done earlier using the trace as the inner product. Hence the ij^th entry of $G([N_1, \dots, N_q])$ is $< N_i, N_j >= Tr(N_i^T N_j)$. Again the grammian is equal to the determinant $det(N)$, and the properties of the grammian readily provide the desired invertibility conditions.

Again we show that the problem always has a solution, i.e. we can always replace mode string $\bar{\sigma}$ with σ_{q+1} while minimizing the error J_2. Moreover, the error is zero if there are n^2 independent basis matrices N_i's from the q available modes, where $N_i = BK_i$. To see this, suppose we have zero independent basis matrices from the q available modes (i.e. $N_i = 0$ for all q modes), then the system is autonomous. Hence, any combination of the q modes will result in the same trajectory and the problem has infinitely many solutions. Now if there are k independent basis matrices(where $k \in \{1, 2, ..., n^2\}$), then we can let the new mode be a linear combination of the k modes corresponding to these k independent basis matrices. In this case the matrix N defined above will be invertible as noted earlier, and this subproblem of replacing the mode string as a combination of these k modes has an unique solution.

Note however, that the solution to the original problem of obtaining a new mode as a linear combination of the q existing modes may not be unique. Since $N_i \in \mathbb{R}^{n\times n}$, we cannot have more than n^2 independent basis. Hence we have covered all the cases, and the problem always has a solution. Moreover if we have n^2 independent basis matrices, these matrices can span $\mathbb{R}^{n\times n}$, and thus any state x_f is reachable with an appropriate choice of coefficient vector $\vec{\alpha}$, independent of the initial condition x_0. In particular, if $\vec{\alpha}^*$ chosen to minimize the error $\| \Phi_q(n_q)\Phi_{q-1}(n_{q-1}) \cdots \Phi_1(n_1) - \Phi_{q+1}(n_1) \|_F^2$ and we have n^2 independent basis, we obtain $\Phi_{q+1}(n_1) = (A - BK_{q+1}) = \Phi_q(n_q)\Phi_{q-1}(n_{q-1}) \cdots \Phi_1(n_1)$, and the error is zero.

3.2 Continuous Time LTI Systems

Consider the following autonomous linear system:

$$\dot{x}(t) = \begin{cases} A_1 x(t) & \text{if } t \in [0, \frac{T}{2}] \\ A_2 x(t) & \text{if } t \in [\frac{T}{2}, T] \end{cases}. \quad (18)$$

Again $x \in \mathbb{R}^n$ and $x(0) = x_0$ is given, then the evolution of x is given as follows:

$$x(0) = x_0$$

$$x(\tfrac{T}{2}) = e^{A_1 \frac{T}{2}} x_0$$
$$x(T) = e^{A_2 \frac{T}{2}} x(\tfrac{T}{2}) = e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0$$

Now suppose we want to find a new A such that

$$x(T) = e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0 \approx e^{AT} x_0. \tag{19}$$

One way of obtaining A is through the use of the well known Campbell-Baker-Hausdorff (CBH) formula, which can be stated as follows:

Campbell-Baker-Hausdorff (CBH) Formula For any two matrices X, Y sufficiently close to 0 , there exists a matrix $Z \in L(X, Y)$ such that $e^Z = e^X e^Y$. Moreover, Z can be explicitly expressed in the Dynkin form as: $Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] + \frac{1}{12}[Y, [Y, X]] + \ldots$, where $[X, Y] = XY - YX$ is the matrix commutator.

Since CBH formula gives an infinite series, we have to be concerned about the convergence when applying the formula. The convergence of the CBF formula has been well studied [31, 26], and it is shown that the Dynkin series converges for matrices X, Y if there is a Lie norm for which

$$\| X \|_{Lie} + \| Y \|_{Lie} \leq \log(2). \tag{20}$$

Here $\| \cdot \|_{Lie}$ denotes the Lie norm, which is a norm on matrices compatible with Lie multiplication, i.e.

$$\| [X, Y] \|_{Lie} \leq \| X \|_{Lie} \| Y \|_{Lie} .$$

Clearly if T is sufficiently small, then $\| A_i \frac{T}{2} \|_{Lie}$ will meet the bound above (20) for $i = 1, 2$. In this case we should be able to approximate this result by using finite number elements from the Lie algebra.

Using CBH formula it is clear that,

$$\begin{aligned} A &= \tfrac{1}{2}A_1 + \tfrac{1}{2}A_2 + \tfrac{T}{4}[A_1, A_2] + \tfrac{T^2}{8}[A_1, [A_1, A_2]] + \ldots \\ &\equiv \tfrac{1}{2}A_1 + \tfrac{1}{2}A_2 + \tfrac{T}{4}[A_1, A_2] + \Delta(T^2), \end{aligned} \tag{21}$$

where $\Delta(T^2)$ is the remaining part of the series which is polynomial in T of degree greater than T^2. Now let us denote $\tilde{A} = \frac{1}{2}A_1 + \frac{1}{2}A_2 + \frac{T}{4}[A_1, A_2]$, we will show that $\| e^{\tilde{A}+\Delta(T^2)} - e^{\tilde{A}} \|$ is bounded by $o(T^2)$. Hence $x(T) \approx e^{\tilde{A}T}$ for a small enough T. First, note the following expression derived in [14],

$$e^{A+\Delta} - e^A = \int_0^1 e^{(1-\tau)A} \Delta e^{\tau A} d\tau + o(\| \Delta \|). \tag{22}$$

Hence, by manipulating (22) we obtain

$$\| e^{A+\Delta} - e^A \| \leq \| \int_0^1 e^{(1-\tau)A} \Delta e^{\tau A} d\tau \| + o(\| \Delta \|)$$

$$
\begin{aligned}
&\leq \int_0^1 \parallel e^{(1-\tau)A}\Delta e^{\tau A} \parallel d\tau + o(\parallel \Delta \parallel) \\
&\leq \int_0^1 e^{\|A\|} \parallel \Delta \parallel e^{\|A\|} d\tau + o(\parallel \Delta \parallel) \\
&= e^{2\|A\|} \parallel \Delta \parallel + o(\parallel \Delta \parallel) \qquad (23)
\end{aligned}
$$

So in our case (23) is reduced to

$$\parallel e^{\tilde{A}+\Delta(T^2)} - e^{\tilde{A}} \parallel \leq e^{2\|\tilde{A}\|} o(T^2). \qquad (24)$$

We have thus shown that A given by the CBH formula can be approximated by $\tilde{A}$ for a small enough T. Of course we can approximate A by using higher-order Lie brackets to obtain a better approximation if desired. Shortly we will compare this approach with the general approach presented in the next section, which relies on the calculus of variations.

4 General Framework For Mode Augmentation

Consider the following system:

$$\dot{x}(t) = \begin{cases} f_1(x(t)) & \text{if } t \in [0, \tau] \\ f_2(x(t)) & \text{if } t \in [\tau, T] \end{cases}, \qquad (25)$$

where $x \in \mathbb{R}^n$ and $x(0) = x_0$ is given. Here $f_i : \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, 2$, are continuously differentiable functions. Now assume we have a set of continuously differentiable functions $g_i : \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, 2, \ldots, N$, and let

$$\dot{z}(t) = \sum_{i=1}^{N} \alpha_i g_i(z(t)), \text{where } \alpha \in \mathbb{R},\ z \in \mathbb{R}^n, \text{ and } z(0) = x_0. \qquad (26)$$

We want to choose $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ so that the cost function

$$J(\vec{\alpha}) = \int_0^T L(x(t), z(t))dt + \psi(x(T), z(T)) \qquad (27)$$

is minimized, where L and ψ are continuously differentiable in their second argument. Note that for our problem, $g_i = \delta_i(f_1, f_2)$, and that $L(x(t), z(t))$ may be 0 for all t if we are only concerned with the final position, but we will derive the solution to this more general problem formulation.

By adding the constraint with a co-state $\lambda(t)$ to (27), we obtain

$$\tilde{J}(\vec{\alpha}) = \int_0^T \Big[L(x(t), z(t)) + \lambda(t)\Big(\sum_{i=1}^{N} \alpha_i g_i(z(t)) - \dot{z}(t) \Big) \Big] dt + \psi(x(T), z(T)). \qquad (28)$$

Note above that $\tilde{J}(\vec{\alpha})$ denotes the unperturbed cost. Now we perturb (28) in such a way that $\vec{\alpha} \to \vec{\alpha} + \epsilon\vec{\theta}_k$, where $\vec{\theta}_k = [0, \ldots, \theta_k, \ldots, 0]^T$ (note the k^{th}

entry is θ_k and all other entries are 0's), and $\epsilon << 1$, then $z \to z + \epsilon\eta$ is the resulting variation in $z(t)$. Note that above, we dropped the argument t when referring to $z(t)$ and will continue this convention in the following development for compactness with the implicit understanding that x, z and λ are functions of t. Now the perturbed cost is given by

$$\tilde{J}(\vec{\alpha} + \epsilon\vec{\theta_k}) = \int_0^T \Big[L(x, z+\epsilon\eta)) + \lambda\Big(\alpha_1 g_1(z+\epsilon\eta) + \alpha_2 g_2(z+\epsilon\eta) + \ldots + \\ +(\alpha_k + \epsilon\theta_k) g_k(z+\epsilon\eta) + \ldots + \alpha_N g_N(z+\epsilon\eta) - \\ -\dot{z}(t) - \epsilon\dot{\eta}\Big)\Big] dt\psi(x(T), (z+\epsilon\eta)(T)). \tag{29}$$

Hence the Gateaux (also referred to as directional) derivative of $\tilde{J}$ in the direction of $\vec{\theta_k}$ is

$$\begin{aligned}\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) &= \lim_{\epsilon\to 0}\frac{\tilde{J}(\vec{\alpha}+\epsilon\vec{\theta_k}) - \tilde{J}(\vec{\alpha})}{\epsilon} \\ &= \int_0^T \Big[\frac{\partial L}{\partial z}\eta + \sum_{i=1}^N \lambda\alpha_i \frac{\partial g_i}{\partial z}\eta + \lambda\theta_k g_k(z) - \lambda\dot{\eta}\Big] dt + \frac{\partial\psi}{\partial z}\eta(T) \end{aligned} \tag{30}$$

Now by integrating $\lambda\dot{\eta}$ in (30) by parts and rearranging terms, we obtain

$$\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \int_0^T \Big[\frac{\partial L}{\partial z} + \lambda\sum_{i=1}^N \alpha_i\frac{\partial g_i}{\partial z} + \dot{\lambda}\Big]\eta dt + \\ +\theta_k\int_0^T \lambda g_k(z)dt - \Big[\lambda\eta\Big]_0^T + \frac{\partial\psi}{\partial z}\eta(T) \tag{31}$$

Note that $\eta(0) = 0$ since $z(0) = x(0) = x_0$. Now choose

$$\lambda(T) = \frac{\partial\psi}{\partial z} \tag{32}$$

$$\dot{\lambda}(t) = -\frac{\partial L}{\partial z}(x,z) - \lambda(t)\sum_{i=1}^N \alpha_i\frac{\partial g_i}{\partial z}(z) \tag{33}$$

With this choice of the co-state $\lambda(t)$, which can be solved by integrating (33) backwards with initial condition (32), we obtain

$$\nabla_{\vec{\theta}_k}\tilde{J}(\vec{\alpha}) = \Big[\int_0^T \lambda g_k(z(t))dt\Big]\theta_k \tag{34}$$

Finally, note that (34) gives access to the partial derivative $\frac{\partial\tilde{J}}{\partial\alpha_k}$ since we know that

$$\nabla_{\vec{\theta}}\tilde{J}(\vec{\alpha}) = \frac{\partial\tilde{J}}{\partial\alpha_1}\theta_1 + \ldots + \frac{\partial\tilde{J}}{\partial\alpha_N}\theta_N, \tag{35}$$

where $\vec{\theta} = [\theta_1, \ldots, \theta_N]^T$. Hence using (34) , (35), and the fact that α_k's are independent of each other, we deduce that

$$\frac{dJ}{d\alpha_k} = \int_0^T \lambda(t) g_k(z(t)) dt. \tag{36}$$

We summarize these results in the following theorem:

Theorem *Given a function $x(t) \in \mathbb{R}^n$ and a set of continuously differentiable functions $g_i : \mathbb{R}^n \to \mathbb{R}^n$ for $i = 1, 2, \ldots, N$, with $z(t) \in \mathbb{R}^n$ given by (26), an extremum to the cost function*

$$J(\vec{\alpha}) = \int_0^T L(x(t), z(t)) dt + \psi(x(T), z(T))$$

is attained when the control vector $\vec{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$ is chosen such that

$$\frac{dJ}{d\alpha_k} = \int_0^T \lambda(t) g_k(z(t)) dt = 0 \quad \textit{for } k = 1, 2, \ldots, N,$$

where the co-state $\lambda(t)$ is chosen as follows:

$$\lambda(T) = \frac{\partial \psi}{\partial z}$$
$$\dot{\lambda}(t) = -\frac{\partial L}{\partial z}(x, z) - \lambda(t) \sum_{i=1}^{N} \alpha_i \frac{\partial g_i}{\partial z}(z).$$

The motivation for obtaining an expression for the gradient (36) is that we can now employ a gradient descent method. The following numerical algorithm is proposed:

At each iteration n, where $\vec{\alpha}^{(n)}$ is the current vector of control variables, we follow these steps :

1. Compute the approximation function $z(t)$ forward in time from 0 to T using (26).

2. Compute the co-state $\lambda(t)$ backward in time from T to 0 using (32) and (33).

3. Compute the gradient $\nabla \tilde{J}(\vec{\alpha}^{(n)}) = \left[\frac{\partial \tilde{J}}{\partial \alpha_1^{(n)}}, \ldots, \frac{\partial \tilde{J}}{\partial \alpha_N^{(n)}} \right]^T$ using (36).

4. Update the control variables as follow :

$$\vec{\alpha}^{(n+1)} = \vec{\alpha}^{(n)} - \gamma^{(n)} \nabla \tilde{J}(\vec{\alpha}^{(n)})$$

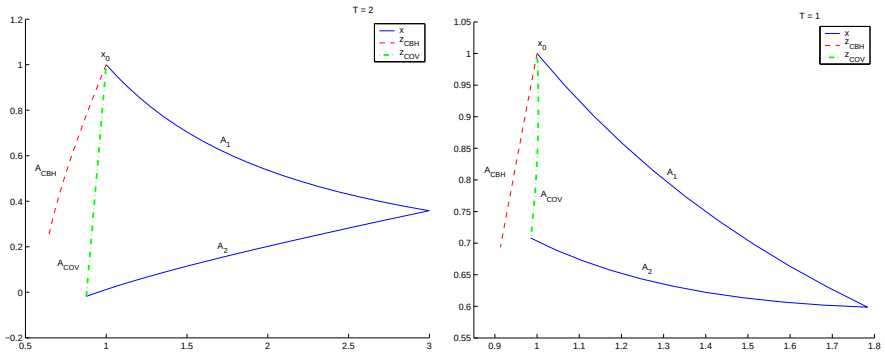
Note that the choice of the stepsize $\gamma^{(n)}$ can be critical for the method to converge. An efficient method among others is the use of Armijo's algorithm presented in [2]. Because of the non-convex nature of the cost function J, this gradient descent algorithm will only converge to a local minimum. Hence the attainment of a "good" local minimum can be quite dependent on the choice of a "good" initial guess for the control variables. However, the method presented here still offers significant reductions in the cost function. The association of such a local method with heuristic strategies in order to find a global minimum is not investigated here.

4.1 Linear Example

Here we consider a specific example of a continuous time LTI system and compare the method described here to the solution derived earlier using the CBH formula. Let

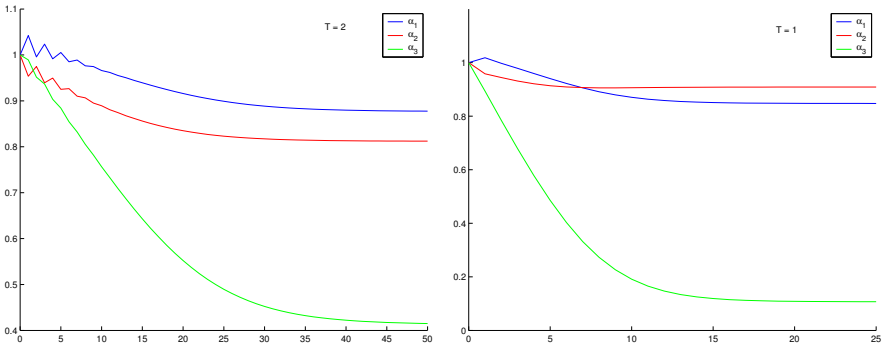
$$\dot{x}(t) = \begin{cases} \begin{pmatrix} 1 & 0.3 \\ 0 & -1 \end{pmatrix} x(t) & \text{if } t \in [0, \frac{T}{2}] \\ \begin{pmatrix} -1.2 & 0.1 \\ -0.3 & 1 \end{pmatrix} x(t) & \text{if } t \in [\frac{T}{2}, T] \end{cases} . \tag{37}$$

Suppose $x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and let's derive a new matrix A_{new} that transfers the system from x_0 to $x(T) \approx e^{A_2 \frac{T}{2}} e^{A_1 \frac{T}{2}} x_0$ in time T without the switch at time $\frac{T}{2}$. Figure 3 (a) shows the trajectories obtained using the first-order approximation of the CBH formula and the corresponding calculus of variation approximation using $\hat{A}_{new} = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_2 [A_1, A_2]$ with $T = 2$. Note that the calculus of variations approach obtained a virtually perfect match, while the CBH approximation was not very accurate. In this case $\vec{\alpha}^* = (0.8763, 0.8112, 0.4134)^T$, hence $\hat{A}_{new} = 0.8763 A_1 + 0.8112 A_2 + 0.4134 [A_1, A_2]$ as opposed to $\tilde{A}_{new} = 0.5 A_1 + 0.5 A_2 + 0.5 [A_1, A_2]$, given by the CBH formula. The CBH formula expectedly provides a better approximation when $T = 1$ (i.e. for a smaller T) as shown in Figure 3 (b). The evolution of $\vec{\alpha}$ in the steepest descent algorithm for both cases ($T = 2, 1$) is shown in Figure 4. Here $\gamma^{(n)} = 0.05$ for $T = 2$ and $\gamma^{(n)} = 0.2$ for $T = 1$ is the step size at iteration n, and it should be noted that the algorithm converges quickly. Observe that the calculus of variations result depends on the initial condition x_0 and hence $\vec{\alpha}^*$ will vary as x_0 varies, however the CBH formula provides global results that are independent of the initial condition x_0.



(a) (b)

Figure 3: Comparison of the CBH and COV methods. (a) $T = 2$, in this case $\| x(T) - z_{CBH}(T) \| = 0.3459$, while $\| x(T) - z_{COV}(T) \| = 1.81 \cdot 10^{-5}$ (b) $T = 1$, in this case $\| x(T) - z_{CBH}(T) \| = 0.0736$, while $\| x(T) - z_{COV}(T) \| = 5.84 \cdot 10^{-4}$



(a) (b)

Figure 4: The evolution of $\vec{\alpha}$ for (a) T = 2 and (b) T = 1, note here that the step size $\gamma^{(n)} = 0.05$ and 0.2 respectively.

4.2 Robotics Example

Consider again the problem of steering a unicycle. The dynamics for this system are

$$\begin{aligned} \dot{x} &= v\cos(\phi), \\ \dot{y} &= v\sin(\phi), \\ \dot{\phi} &= \omega. \end{aligned} \tag{38}$$

In the system above (x, y) are the Cartesian coordinate of the center of the unicycle and ϕ is its orientation with respect to the x-axis. Assume that v is constant and ω is the control variable. Given that the system initially has two

behaviors, namely "go-to-goal" and "avoid-obstacle", the feedback mappings associated with each behavior are as follow:

$$\kappa_g(x, y, \phi) = \omega_g = C_g(\phi_g - \phi), \tag{39}$$

$$\kappa_o(x, y, \phi) = \omega_o = C_o(\pi + \phi_o - \phi). \tag{40}$$

Note here that C_g and C_o are the gains associated with each behavior, and ϕ_g and ϕ_o are the angles to the goal and nearest obstacle respectively. Both of these angles are measured with respect to the x-axis and can be expressed as

$$\phi_g = \arctan(\frac{y_g - y}{x_g - x}) \text{ and } \phi_o = \arctan(\frac{y_{obs} - y}{x_{obs} - x}), \tag{41}$$

where (x_g, y_g) and (x_{obs}, y_{obs}) are the Cartesian coordinates of the goal and the nearest obstacle respectively. We also have a set of three interrupts, $\xi_{1,2,3}(x)$, that trigger at three different distances away from the nearest obstacle (x_{obs}, y_{obs}), and all three interrupts always trigger at the goal (x_g, y_g). Hence the total number of available modes is six, i.e. $card(\Sigma) = 6$. The problem then is to plan a path from an initial state (x_0, y_0, ϕ_0) to an open ball around (x_g, y_g) given the set of modes above while minimizing the string length of the control program (i.e. number of switches) along with the total distance travelled.

Given this set of modes, we begin by exploring the reachable space and then perform reinforcement learning to find the optimal path as described earlier. The resulting path for our problem is shown in Figure 5. The optimal control sequence in this case is $\bar{\sigma}^* = (\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_3)(\kappa_g, \xi_1)(\kappa_o, \xi_1)(\kappa_g, \xi_1)$. So clearly $(\kappa_o, \xi_3)(\kappa_g, \xi_1)$ is repeated often in the optimal control program, thus it would be beneficial to replace it with a single mode (κ_n, ξ_1), where $\kappa_n = \alpha_g \kappa_g + \alpha_o \kappa_o$. Using the variational techniques given here, it is found that $\alpha_g^* = 0.211$ and $\alpha_o^* = 0.801$. Now we recalculate the optimal path with the new feedback mapping $\kappa_n(x)$ and again the three existing interrupts for its termination added to the mode set. The resulting path is shown in Figure 6 and the optimal control sequence is given by $\tilde{\sigma}^* = (\kappa_g, \xi_1)(\kappa_n, \xi_3)(\kappa_g, \xi_1)$. The augmentation of the motion alphabet results in great improvement in terms of the optimal mode sequence and the resulting optimal trajectory. Although we only designed the new feedback map to "merge" two modes, the overall affect of adding the new modes reduced the size of the control program from $|\sigma^*| = 7$ to $|\tilde{\sigma}^*| = 3$. Moreover, the complexity of the control program is reduced from $7 \cdot log_2(6) = 18.0947$ to $3 \cdot log_2(9) = 9.5098$.

5 Conclusions

In this paper we address two issues that arise when designing multi-model control strategies. The first issue concerns the problem of concatenating given control modes so as to minimize a particular cost. This problem is solved by a transformation of the original continuous-time, infinite state-and-input space problem to a finite discrete-time problem to which standard reinforcement learning techniques can be applied.

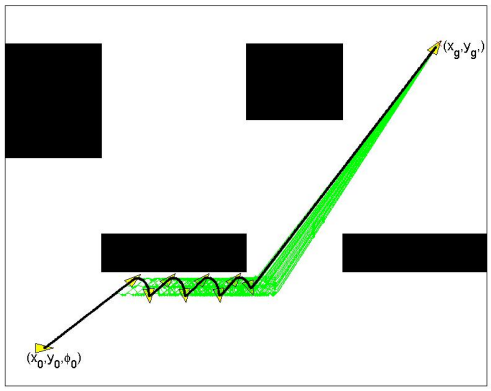


Figure 5: The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the available set of modes.

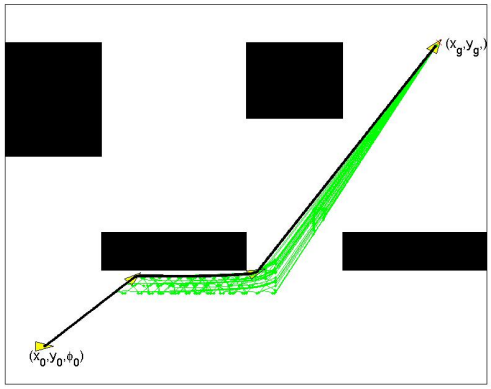


Figure 6: The estimated reachable set along with the optimal path (thick) to drive a unicycle from x_0 to x_g using the augmented set of modes.

The second issue concerns the problem of mode generation. Given a control program (i.e. a string of control modes) containing recurring sub-strings, the

problem under investigation is how to translate such strings into one single mode in an optimal manner. Thus, resulting in the reduction of the length of the control program, and consequently a reduction in the specification complexity of the program. The solution to this problem is given for a number of different systems ranging from discrete-time linear control systems to fully nonlinear systems. A number of examples are presented that illustrate the numerical viability of the proposed approach.

References

[1] R.C. Arkin. *Behavior Based Robotics*. The MIT Press, Cambridge, MA, 1998.

[2] L. Armijo. Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives. *Pacific Journal of Mathematics*, Vol. 16, ppm. 1-3, 1966.

[3] J. A. Bathurin. *Lectures on Lie algebras, Akademie Verlag*, Berlin, 1978.

[4] A. Bhatia, and E. Frazzoli. Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems. *Hybrid Systems: Computation and Control. Springer-Verlag*, 2004.

[5] A. Bicchi, A. Marigo, and B. Piccoli. On the reachability of quantized control systems. *IEEE Transactions on Automatic Control*, 4(47):546-563, April 2002.

[6] A. Bicchi, A. Marigo, and B. Piccoli. Encoding steering control with symbols. *IEEE International Conference on Decision and Control*, pages 3343-3348, 2003.

[7] S.J. Bradtke, B.E. Ydstie, and A.G. Barto. Adaptive linear quadratic control using policy iteration. *American Control Conference*, pages 3475-3479, 1994.

[8] R.W. Brockett. On the Computer Control of Movement. emphProceedings of the 1988 IEEE Conference on Robotics and Automation, pp. 534540, New York, April 1988.

[9] P. Cheng, Z. Shen, and S. M. LaValle. Using randomization to find and optimize feasible trajectories for nonlinear systems. *Proc. Annual Allerton Conference on Communications, Control, Computing*, pages 926–935, 2000.

[10] L. Crawford, and S.S. Sastry. Learning Controllers for Complex Behavioral Systems. *Neural Information Processing Systems Tenth Annual Conference(NIPS 96)*, 1996.

[11] M. Egerstedt. On the Specification Complexity of Linguistic Control Procedures. *International Journal of Hybrid Systems*, Vol. 2, No. 1-2, pp. 129-140, March & June, 2002.

[12] M. Egerstedt, and D. Hristu-Varsakelis. Observability and Policy Optimization for Mobile Robots. *IEEE Conference on Decision and Control, Las Vegas, NV*, Dec. 2002.

[13] 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

[14] T. T. Georgiou. Relative Entrophy and the multi-variable multi-dimesional Moment Problem. submitted to *IEEE Transaction on IT* CLN 04-326. Revised Dec, 2004.

[15] T. A. Henzinger. The theory of hybrid automata. *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society Press, 1996, pp. 278-292.

[16] D. Hristu-Varsakelis, M. Egerstedt, and P.S. Krishnaprasad. On The Structural Complexity of the Motion Description Language MDLe. *IEEE Conference on Decision and Control*, Dec. 2003.

[17] T. Jaakkola, M.I. Jordan, and S.P. Singh. On the Convergence of stochastic iterative dynamic programming algorithms. *Neural Computation* 6(6), 1994.

[18] L.P. Kaebling, M.L. Littman, and A.R. Cassandra. Learning Policies for Partially Observable Environments: Scaling Up. *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[19] L.P. Kaebling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal Of Artificial Intelligence Research*, 1996.

[20] G. Lafferriere, H. J. Sussmann. A Differential Geometric Approach to motion Planning. *Nonholonomic Motion Planning, Z. Li and J.F. Canny, Eds, Kluwer Academic Publishers*, pp 235-270, 1993

[21] S. M. LaValle. From dynamic programming to RRTs: Algorithmic design of feasible trajectories. *Control Problems in Robotics*, pages 19–37. Springer-Verlag, 2002.

[22] Lygeros, J., Tomlin, C., and Sastry, S. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), pp 349-370, 1999

[23] T. Mehta and M. Egerstedt. Learning Multi-Modal Control Programs. *Hybrid Systems: Computation and Control, Springer-Verlag*, March 2005.

[24] K. Morgansen, and R.W. Brockett. Optimal Regulation and Reinforcement Learning for the Nonholonomic Integrator. *Proceedings of the American Control Conference*, pp. 462-6, June 2000

[25] G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035-2047, December 2003.

[26] W. Rossmann. *Lie Groups: An Introduction Through Linear Groups, Oxford University Press*, Chapter 1.3, 2002.

[27] H. J. Sussmann. Lie brackets and local controllabilty: a sufficient condition for scalar input systems. *SIAM J. Control and Optimization*, 21(5): 686-713, 1983.

[28] R.S. Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. *Neural Information Processing Systems 8*, 1996.

[29] R.S. Sutton, and A.G. Barto. *Reinforcement Learning, An Introduction.* MIT Press, Cambridge, MA, 1998.

[30] P. Tabuada and G. Pappas. Model Checking LTL over Controllable Linear Systems is Decidable. *Hybrid Systems: Computation and Control*, Springer-Verlag, Prague, Czech Republic, 2003.

[31] R. C. Thompson. Lecture 10 : Part I - Convergence domains of the Campbell Baker Hausdorff formula, *John Hopkins Lecture Notes*, 1988.

[32] J.N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16(3), 1994.

[33] C.J.C.H. Watkins, and P. Dayan. Q-learning. *Machine Learning* 8(3/4):257-277, May 1992.