Hybrid Optimal Control Examples

This document illustrates the performance of our approach using several examples. Before proceeding, we begin by describing the numerical implementation of our algorithm. First, for each of the systems described below, we denote the range space of control inputs in mode i as U_i. In the implementation, we can always define $U := \prod_{i \in \mathcal{I}} U_i$ without causing any problems. Second, for each of the optimal control problems, we implement our algorithm using the MOSEK [1] numerical solver in MATLAB and generate a polynomial feedback control law. Third, the trajectory is obtained by plugging the (saturated) polynomial control law back into the system dynamics in each mode and simulating forward using a standard ODE solver with event detection in MATLAB. Once the trajectory hits a guard, another ODE solver is initialized at a new point given by the associated reset map, and the simulation continues in the same way until terminal condition is satisfied. Next, for the sake of comparison, all the examples are solved either analytically (when possible) or using GPOPS-II [2] by iterating through a finite set of of possible transitions. Notice that in this latter instance we must fix the possible sequences of transitions and provide an initial condition, since existing numerical hybrid optimal control algorithms require this information. Finally, all of our experiments are performed on an Intel Xeon, 20 core, 2.60 GHz, 128 GB RAM machine.

A. Hybridized Double Integrator

The double integrator is a two-state, single-input linear system. Even though a standard double integrator is a non-hybrid system, we may hybridize it by dividing the domain into two parts, and defining an identity reset map between them as described in Table I and Table II.

TABLE I: Vector fields and domains of each of the modes of the hybridized double integrator

Mode	$i = 1$	$i = 2$
Dynamics	$\dot{x}(t) = \begin{bmatrix} x_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$	$\dot{x}(t) = \begin{bmatrix} x_2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$
X_i	$\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 0.3\}$	$\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \geq 0.3\}$
U_i	$[-1, 1]$	$[-1, 1]$

TABLE II: Guards and reset maps of the hybridized double integrator. The rows are modes in which a transition originates, and the columns are modes to which the transition goes.

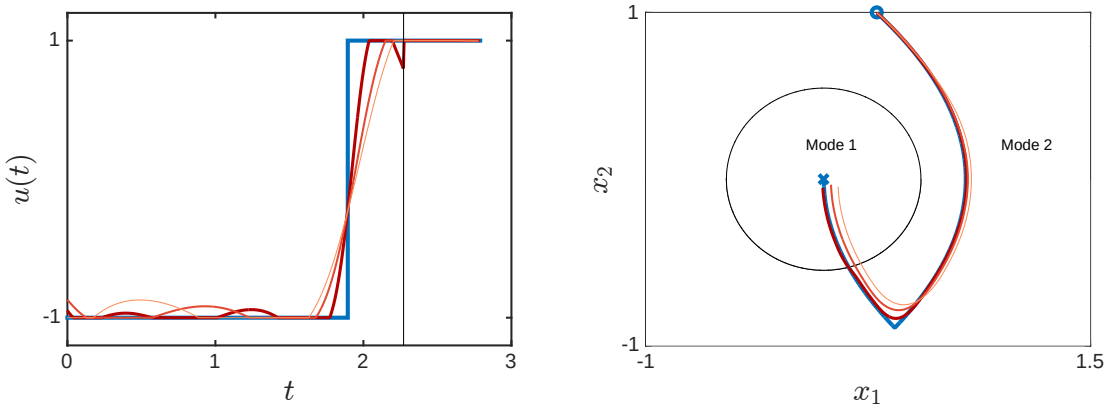
	Mode 1	Mode 2
Mode 1	N/A	N/A
Mode 2	$S_{(2,1)} = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 = 0.3\}$ $R_{(2,1)}(x) = x, \quad \forall x \in S_{(2,1)}$	N/A

Note that X_2 is not compact, but we may impose the additional constraint $\|x(t)\|_\infty \leq N$ on X_2 for some large N [3, Section 5.1]. However, this additional constraint is not enforced in the numerical implementations. We first consider the following minimum time problem: drive the system to the point $(0, 0)$ beginning from $x_0 = (0.3, 1) \in X_1$ in minimum time. We assume the minimum time needed is less than 5 and the problem is set up according to Table III.

TABLE III: Optimal control problem setup of the hybridized double integrator

	$i = 1$	$i = 2$
h_i	1	1
H_i	0	0
x_0	$(0.3, 1) \in X_1$	N/A
X_{T_i}	$\varnothing$	$\{(0, 0)\} \subset X_2$
T_0	5	

For this system, the optimal admissible pair is analytically computable, which is used as ground truth and compared to the result of our method with degrees of relaxation $k = 6$, $k = 8$, and $k = 12$ in Figure 1. The polynomial control law is saturated so that its value is in U for all time. The cost and computation time are also compared in Table V.



(a) Control action

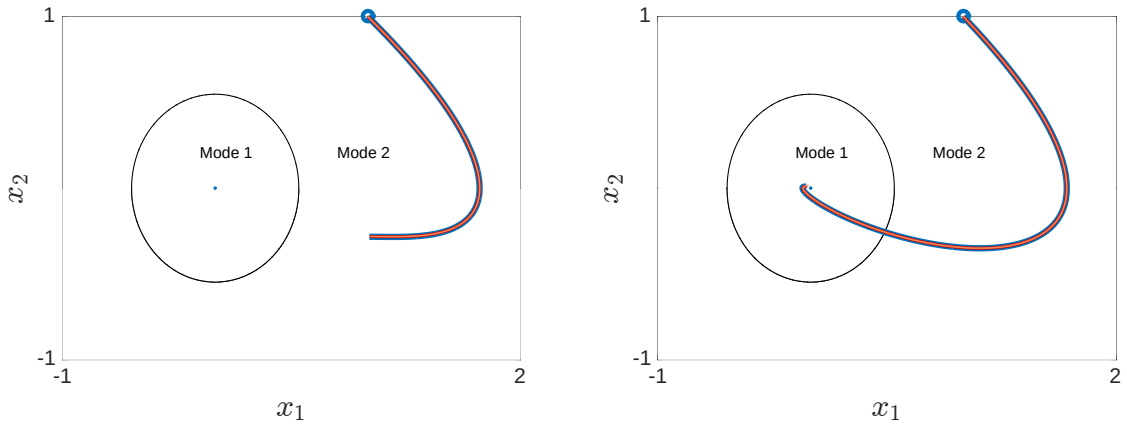
(b) Trajectory

Fig. 1: An illustration of the performance of our algorithm on a free final time version of the hybridized double integrator problem. The blue circle indicates the given initial point x_0, and the blue cross shows the target set. The blue line is the analytically computed optimal control, while the red lines of various saturation correspond to control actions generated by our method. When the simulated trajectory does not pass through $(0,0)$ perfectly, the simulation terminates when the closest point is reached. As the saturation of the color in the illustration increases the corresponding degree of relaxation increases between $2k = 6$ to $2k = 8$ to $2k = 12$. Figure 1a depicts the control action whereas Figure 1b illustrates the resultant trajectory when forward simulated through the system. The moment of transition from mode 2 to mode 1 is indicated by a vertical black solid line in Figure 1a.

Next, we consider an Linear Quadratic Regulator (LQR) problem on the same hybridized double integrator system, where the goal is to drive the system state towards $(0,0)$ while keeping the control action small for all time $t \in [0,T]$. The problem is set up according to Table IV. To further illustrate we are able to handle different number of modes visited, two cases where $T = 5$ and $T = 15$ are considered. For comparison, the LQR problem is also solved by a standard finite-horizon LQR solver in the non-hybrid case, which we refer to as the ground truth. The results are compared in Figure 2 and Table V with degrees of relaxation $2k = 6$, $2k = 8$, and $2k = 12$.

TABLE IV: Problem setup of the hybridized double integrator LQR problem

	$i = 1$	$i = 2$
h_i	$x_1^2 + x_2^2 + 20 \cdot u^2$	$x_1^2 + x_2^2 + 20 \cdot u^2$
H_i	0	0
x_0	$(1,1) \in X_1$	N/A
X_{T_i}	$\{(x_1,x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 0.3\} = X_1$	$\{(x_1,x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \geq 0.3\} = X_2$
T	5 or 15	



(a) Trajectory, T = 5

(b) Trajectory, T = 15

Fig. 2: An illustration of the performance of our algorithm on LQR version of the hybridized double integrator problem. The blue circles indicate the given initial point x_0, and the blue dots show the point $(0, 0)$. The blue lines are the analytically computed optimal control, while the red lines of various saturation correspond to control actions generated by our method. As the saturation of the color increases the corresponding degree of relaxation increases between $2k$ = 6 to $2k$ = 8 to $2k$ = 12. Figure 2a shows the trajectories when T = 5, and Figure 2b shows the trajectories when T = 15.

TABLE V: Results of the hybridized double integrator examples

		Computation time	Cost returned from optimization	Cost returned from simulation
Minimum time problem with T_0 = 5	$2k$ = 6	3.1075[s]	2.7781	2.7780 [a]
	$2k$ = 8	10.0187[s]	2.7847	2.7845 [a]
	$2k$ = 12	170.9319[s]	2.7868	2.7865 [a]
	Ground truth	N/A	2.7889	N/A
LQR problem with T = 5	$2k$ = 6	2.2299[s]	24.9496	24.9906
	$2k$ = 8	8.1412[s]	24.9496	24.9906
	$2k$ = 12	198.2826[s]	24.9502	24.9906
	Ground truth	N/A	24.9503	N/A
LQR problem with T = 15	$2k$ = 6	2.1965[s]	26.1993	26.3428
	$2k$ = 8	7.7989[s]	26.1993	26.3438
	$2k$ = 12	168.5383[s]	26.1996	26.3435
	Ground truth	N/A	26.2033	N/A

[a]Trajectory does not reach target set perfectly. The simulation terminates when the closest point is reached

B. Dubins Car Model with Shortcut Path

The next example shows our algorithm can work with different dimensions in each mode, and is capable of choosing the best transition sequence. Consider a 2-mode hybridized Dubins Car system with identity reset map. We now add another 1-dimensional mode to the system, and connect it with the other two modes by defining transitions. The vector fields, guards, and reset maps are defined in Table VI and Table VII. In mode 1 and mode 2, the control is $u = (v, \omega)$; In mode 2, the control is $u = v$. Although the dynamics in mode 1 and mode 2 are not polynomials, they are approximated by 2nd-order Taylor expansion around $x = (0, 0, 0)$ in the numerical implementation. We are interested in solving the minimum time problem, where the trajectory starts at $x_0 = (-0.8, 0.8, 0)$ in mode 1, and ends at x, y-position $(0.8, -0.8)$ in mode 2. The optimal control problem is defined in Table VIII.

TABLE VI: Vector fields and domains of the Dubins car model with shortcut path

Mode	$i = 1$	$i = 2$	$i = 3$
Dynamics	$\dot{x}(t) = \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}\cos(x_3(t)) & 0\\ \sin(x_3(t)) & 0\\ 0 & 1\end{bmatrix} u$	$\dot{x}(t) = \begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}\cos(x_3(t)) & 0\\ \sin(x_3(t)) & 0\\ 0 & 1\end{bmatrix} u$	$\dot{x}(t) = 0 + (-1) \cdot u$
X_i	$[-1, 1] \times [0, 1] \times [-\pi, \pi] \subset \mathbb{R}^3$	$[-1, 1] \times [-1, 0] \times [-\pi, \pi] \subset \mathbb{R}^3$	$[-1, 1] \subset \mathbb{R}$
U_i	$[0, 1] \times [-3, 3]$	$[0, 1] \times [-3, 3]$	$[0, 2]$

TABLE VII: Guards and reset maps of the Dubins car model with shortcut path. The rows are modes in which a transition originates, and the columns are modes to which the transition goes.

	Mode 1	Mode 2	Mode 3
Mode 1	$S_{(1,2)} = [-1, 1] \times \{0\} \times [-\pi, \pi]$ $R_{(1,2)}(x) = x, \quad \forall x \in S_{(2,1)}$	N/A	$S_{(1,3)} = [-1, 1] \times \{1\} \times [-\pi, \pi]$ $R_{(1,3)}(x) = 1, \quad \forall x \in S_{(1,3)}$
Mode 2	N/A	N/A	N/A
Mode 3	N/A	$S_{(3,2)} = \{-1\}$ $R_{(3,2)}(x) = (0.6, -0.8, 0),$ $\forall x \in S_{(3,2)}$	N/A

TABLE VIII: Problem setup of Dubins car model with shortcut path

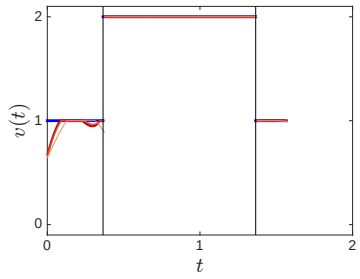
Mode	$i = 1$	$i = 2$	$i = 3$
h_i	1	1	1
H_i	0	0	0
x_0	$(-0.8, 0.8, 0) \subset X_1$	N/A	N/A
X_{T_i}	N/A	$\{0.8\} \times \{-0.8\} \times [-\pi, \pi] \subset X_2$	N/A
T	3		

Notice the transition sequences "1-2" and "1-3-2" are both feasible in this instance according to our guard definition, but direct calculation shows that we may arrive at the target point in less time by taking the "shortcut path" in mode 3. This problem is solved using our algorithm with degrees of relaxation $2k = 6$, $2k = 8$, and $2k = 10$.

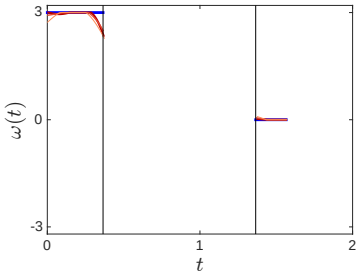
TABLE IX: Results of the Dubins car example with shortcut path

	Computation time	Cost returned from optimization	Cost returned from simulation
$2k$ = 6	83.0224[s]	1.5641	1.5739
$2k$ = 8	1.2115×10^3[s]	1.5647	1.5679
$2k$ = 10	1.3206×10^4[s]	1.5648	1.5703
Ground truth	N/A	1.5651	N/A

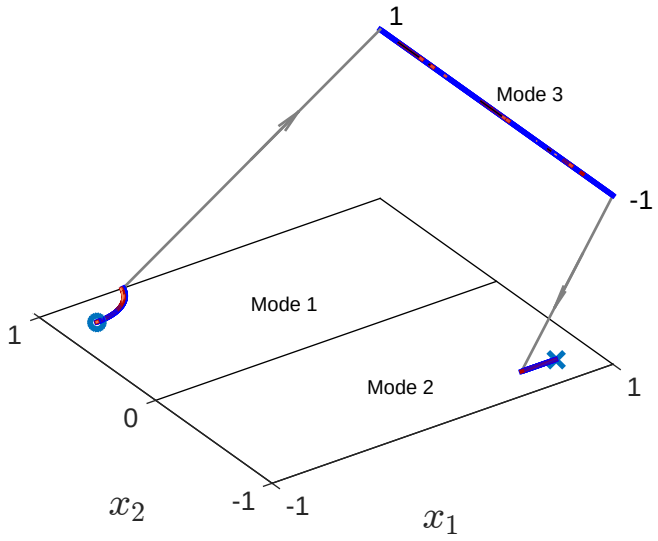
As comparison, we treat the analytically computed optimal control as ground truth, and the results are compared in Figure 3 and Table IX. In this example our algorithm is able to pick the transition sequence "1-3-2" and find a tight approximation to the true optimal solution.



(a) Control action: v



(b) Control action: ω



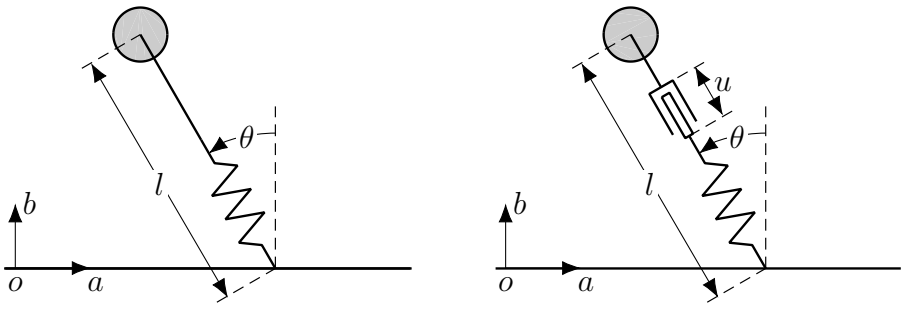
(c) Trajectory

Fig. 3: An illustration of the performance of our algorithm on a minimum time problem of the Dubins car model with shortcut path. The blue circle indicates the given initial point x_0, and the blue cross shows the target set. The blue solid line is the analytically computed optimal control, and the red lines of various saturations are controls generated by our method. As the saturation increases the corresponding degree of relaxation increases between $2k$ = 6 to $2k$ = 8 to $2k$ = 10. Figure 3a, Figure 3b depict the control actions whereas Figure 3c illustrates the corresponding trajectory obtained by forward simulating through the system. The moment of transitions are indicated by vertical black solid lines in Figure3a and Figure 3b.

C. SLIP Model

The Spring-Loaded Inverted Pendulum (SLIP) is a classical model that describes the center-of-mass dynamics of running animals and robots, and has been extensively used as locomotion template to perform control law synthesis on legged robots [4]. Despite its simplicity, an analytical solution to the SLIP dynamics does not exist. We may simulate the system numerically, but the optimal control problem is still difficult to solve if the sequence of transition is not known beforehand.

As is shown in Figure 4a, the SLIP is a mass-spring physical system, modeled as a point mass, M, and a massless spring leg with stiffness k and length l. The dynamics of SLIP consist of two phases: stance phase and flight phase. The stance phase starts when the leg comes into contact with the ground with downward velocity, which we call the *touchdown* event, and ends when the leg extends to full length and leaves the ground, which we call the *liftoff* event. During the stance phase, the inverted pendulum swings forward around the leg-ground contact point, while the spring contracts due to mass momentum and gravitational force. During flight phase, SLIP follows free fall motion where the only external force is the gravity. We also assume the leg angle is reset to some fixed value



(a) SLIP model (b) Active SLIP model

Fig. 4: SLIP model and system variable definition

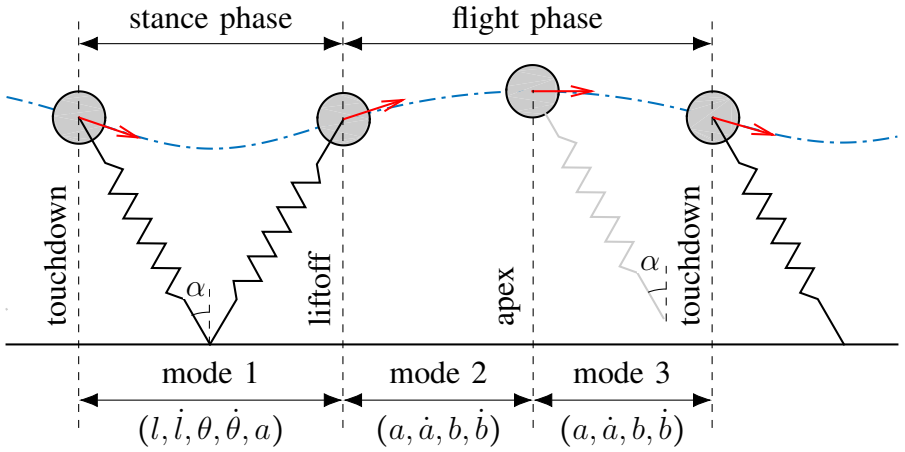


Fig. 5: SLIP locomotion phases and hybrid system modes

α instantaneously once the SLIP enters flight phase, so that the leg angle at the moment of touchdown stays the same. Furthermore, we define the *apex* event to be when the body reaches its maximum height with zero vertical velocity. The touchdown, liftoff, and apex events are illustrated in Figure 5.

In the context of this paper, we are interested in the active SLIP model (Figure 4b), where a mass-less actuator is added to the SLIP leg. During stance phase, the actuator may extend from its nominal position within some range, while during flight phase, the actuator has no effect on the system. The active SLIP can be modeled as a hybrid system with 3 modes, where the liftoff, apex, and touchdown events define the transitions between them, as shown in Figure 5.

TABLE X: State variables of the active SLIP mode

l	leg length	a	horizontal displacement
$\dot{l}$	time derivative of l	$\dot{a}$	time derivative of a
θ	leg angle	b	vertical displacement
$\dot{\theta}$	time derivative of θ	$\dot{b}$	time derivative of b

TABLE XI: Physical parameters of the active SLIP model

	Explanation	Value
M	mass	1
k	spring constant	6
g_0	gravitational acceleration	0.2
l_0	nominal leg length	0.2
α	reset angle in flight phase	$\pi/6$

The behavior of such a system can be fully characterized using 8 variables defined in Table X. In mode 1, we define the system state to be $x = (l, \dot{l}, \theta, \dot{\theta}, a)$; In mode 2 and mode 3, we define the system state to be $x = (a, \dot{a}, b, \dot{b})$. The physical parameters, dynamics, and transitions are defined in Table XI, Table XII, and Table XIII. Again, we use 3rd-order Taylor expansion around $(l_0, 0, 0, 0, 0)$ to approximate the stance phase dynamics with polynomials.

TABLE XII: Vector fields and domains of each of the modes of the active SLIP model

Mode	$i = 1$	$i = 2$	$i = 3$
Dynamics	$\dot{x}(t) = \begin{bmatrix} x_2(t) \\ -\frac{k}{M}(x_1(t) - l_0) - g_0\cos(x_3(t)) \\ x_4(t) \\ -\frac{2x_2(t)x_4(t)}{x_1(t)} - \frac{g_0\sin(x_3(t))}{x_1(t)} \\ -x_2(t)\sin(x_3(t)) - x_1(t)x_4(t)\cos(x_3(t)) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{k}{M} \\ 0 \\ 0 \\ 0 \end{bmatrix} u$	$\dot{x}(t) = \begin{bmatrix} x_2(t) \\ 0 \\ x_4(t) \\ -g_0 \end{bmatrix}$	$\dot{x}(t) = \begin{bmatrix} x_2(t) \\ 0 \\ x_4(t) \\ -g_0 \end{bmatrix}$
X_i	$[0.1, 0.2] \times [-0.3, 0.3] \times [-1, 1] \times [-3, 0] \times [-1, 1] \subset \mathbb{R}^5$	$[-1, 1] \times [0, 0.5] \times [0.15, 0.5] \times [0, 1] \subset \mathbb{R}^4$	$[-1, 1] \times [0, 0.5] \times [l_0\cos(\alpha), 0.5] \times [-1, 0] \subset \mathbb{R}^4$
U_i	$[0, 0.1]$	N/A	N/A

We fix the initial condition, and consider the following two hybrid optimal control problems for the active SLIP: In the first problem, we maximize the vertical displacement b up to time $T = 2.5$. In stance phase, the 1st-order Taylor approximation $b = l\cos(\theta) \approx l$ is used; In the second problem, we define a constant-speed reference trajectory $a(t) = vt - 0.5$ in the horizontal coordinate, then try to follow this trajectory with active SLIP up to time $T = 3$.

TABLE XIII: Guards and reset maps of the active SLIP model. The rows are modes in which a transition originates, and the columns are modes to which the transition goes.

	Mode 1	Mode 2	Mode 3
Mode 1	N/A	$S_{(1,2)} = \{x \in X_1 \mid x_1 = l_0, x_2 \geq 0\}$ $R_{(1,2)}(x) = \begin{bmatrix} x_5 \\ -x_2\sin(x_3) - l_0 x_4 \cos(x_3) \\ l_0\cos(x_3) \\ x_2\cos(x_3) - l_0 x_4 \sin(x_3)) \end{bmatrix}$, $\forall x \in S_{(1,2)}$	N/A
Mode 2	N/A	N/A	$S_{(2,3)} = \{x \in X_2 \mid x_4 = 0\}$ $R_{(2,3)}(x) = x, \quad \forall x \in S_{(2,3)}$
Mode 3	$S_{(3,1)} = \{x \in X_3 \mid x_3 = l_0\cos(\alpha)\}$ $R_{(3,1)}(x) = \begin{bmatrix} l_0 \\ -x_2\sin(\alpha) + x_4\cos(\alpha) \\ \alpha \\ -\frac{x_2}{l_0}\cos(\alpha) - \frac{x_4}{l_0}\sin(\alpha)) \end{bmatrix}$, $\forall x \in S_{(3,1)}$	N/A	N/A

The optimal control problems are defined in Table XIV. Note that these problems are defined such that the optimal transition sequences are different in each instance, and some modes are visited multiple times.

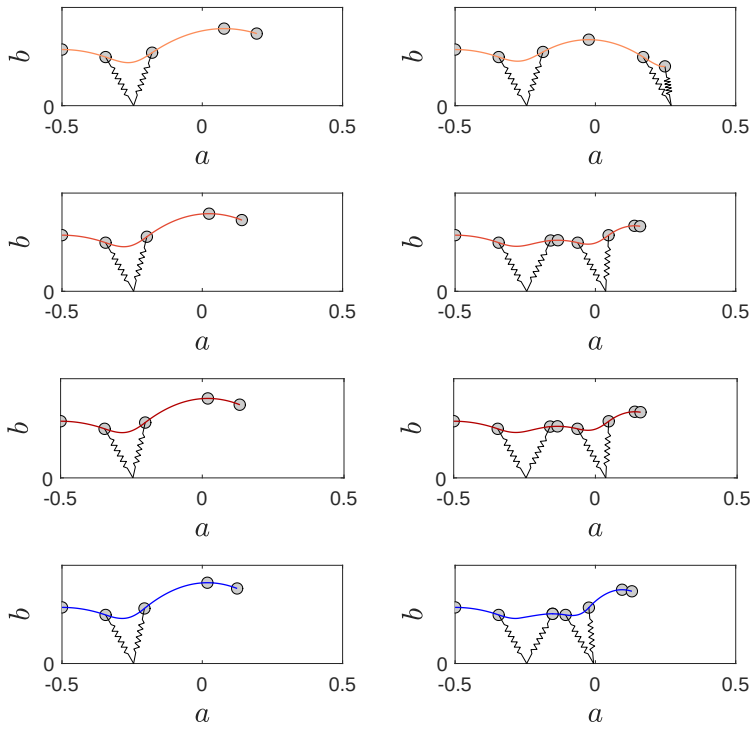
TABLE XIV: Problem setup of SLIP

	Mode	$i = 1$	$i = 2$	$i = 3$
Maximizing vertical displacement	h_i	$-x_1$	$-x_3$	$-x_3$
	H_i	0	0	0
	x_0	N/A	N/A	$(-0.5, 0.3, 0.2, 0) \in X_3$
	X_{T_i}	X_1	X_2	X_3
	T	2.5		
Tracking constant-speed trajectory $a(t) = vt - 1$ with $v = 0.1$	h_i	$(v \cdot t - 0.5 - x_5)^2$	$(v \cdot t - 0.5 - x_1)^2$	$(v \cdot t - 0.5 - x_1)^2$
	H_i	0	0	0
	x_0	N/A	N/A	$(-0.5, 0.3, 0.2, 0) \in X_3$
	X_{T_i}	X_1	X_2	X_3
	T	3		

The optimization problems are solved by our algorithm with degrees of relaxation $2k = 4$, $2k = 6$, and $2k = 8$. For the sake of comparison, the same problems are also solved using GPOPS-II. Since GPOPS-II requires information about the transition sequence, we let the number of transitions to be less than 10, and iterate through all possible transition sequences with GPOPS-II. The results are compared in Figure 6 and Table XV.

TABLE XV: Results of the active SLIP

		Computation time	Cost returned from optimization	Cost returned from simulation
Maximizing vertical displacement	$2k = 4$	42.1805[s]	-0.7003	-0.5480
	$2k = 6$	722.5955[s]	-0.5773	-0.5577
	$2k = 8$	1.2290×10^4[s]	-0.5754	-0.5629
	GPOPS-II	2.6303×10^3[s]	-0.5735	N/A
Tracking constant-speed trajectory $a(t) = vt - 0.5$ with $v = 0.1$	$2k = 4$	28.0788[s]	0.0490	0.22957
	$2k = 6$	503.7731[s]	0.1422	0.18102
	$2k = 8$	1.1433×10^4[s]	0.1493	0.18374
	GPOPS-II	9.3389×10^3[s]	0.1624	N/A



(a) Maximizing vertical displacement (b) Tracking constant speed $v = 0.1$

Fig. 6: An illustration of the performance of our algorithm on active SLIP model. The blue lines are the optimal control computed by GPOPS-II by iterating through all the possible transition sequences, and the red lines of various saturation are controls generated by our method. As the saturation increases the corresponding degree of relaxation increases between $2k = 4$ to $2k = 6$ to $2k = 8$. Figure 6a shows trajectories that maximize vertical displacement, where the optimal solution goes through 3 transitions; Figure 6b shows trajectories that track $v = 0.1$, where the optimal solution goes through 6 transitions.

REFERENCES

[1] M. ApS, *MOSEK MATLAB Toolbox. Release 8.0.0.53.*, 2017. [Online]. Available: http://docs.mosek.com/8.0/toolbox/index.html

[2] M. A. Patterson and A. V. Rao, "Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 1,

p. 1, 2014.

[3] J. B. Lasserre, D. Henrion, C. Prieur, and E. Trélat, "Nonlinear optimal control via occupation measures and lmi-relaxations," *SIAM Journal on Control and Optimization*, vol. 47, no. 4, pp. 1643–1666, 2008.

[4] P. Holmes, R. J. Full, D. Koditschek, and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *Siam Review*, vol. 48, no. 2, pp. 207–304, 2006.